Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2021 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи кібербезпеки силової інфраструктури ЦОД”

Виконав здобувач вищої освіти
ІІ курсу, групи КІ-20М-1,4
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Капкан В.В.
«____» _____ 2021 р.

Керівник проекту
кандидат технічних наук
_____ Сергій СМІРНОВ
«____» _____ 2021 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Рівень вищої освіти *магістр*

Галузь знань ._12 _____*"*Інформаційні технології*"*_____

Спеціальність *123 "*Комп'ютерна інженерія*"*

Освітньо-професійна (освітньо-наукова) програма _*"*Комп'ютерна інженерія*"*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 6 » вересня 2021 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ДРУГИМ (МАГІСТЕРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Капкану Віталію Вікторовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи кібербезпеки силової інфраструктури ЦОД*

2. Керівник роботи *Смірнов Сергій Анатолійович, канд. техн. наук*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 42-13 від 02.08.2021 року

3. Строк подання студентом роботи до захисту *10.12.2021 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи кібербезпеки силової інфраструктури ЦОД*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.	*7. Економічна ефективність розробленої*
2.Перегляд аналогічних існуючих систем.	*програми.*
3. Опис і обґрунтування проектних рішень.	*8. Заходи з охорони праці та техніки безпеки*
4. Етапи програмування системи.	*9. Висновки.*
5. Впровадження системи в промислову експлуатацію	
6. Наукова новизна	

6. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Наукова новизна	*1 аркуш*
Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*
Показники економічної ефективності	*1 аркуш*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Савеленко Г.В.	05.10.2021	14.11.2021
Охорона праці	Оришака О.В.	06.10.2021	16.11.2021

7. Дата видачі завдання « 6 » вересня 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.10.2021 р.	
2.	Постановка задачі, оформлення ТЗ	15.10.2021 р.	
3.	Розробка моделі компонента	20.10.2021 р.	
4.	Розробка структур даних	25.10.2021 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.10.2021 р.	
6.	Програмування алгоритмів	10.11.2021 р.	
7.	Розрахунок економічної ефективності	13.11.2021 р.	
8.	Розрахунки з охорони праці та техніки безпеки	15.11.2021 р.	
9.	Оформлення ПЗ	17.11.2021 р.	
10.	Попередній захист роботи	10.12.2021 р.	

Дата видачі завдання
« 6 » вересня 2021 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2021 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Капкан В.В. Дослідження та програмна реалізація системи кібербезпеки силової інфраструктури ЦОД. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2021.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи кібербезпеки силової інфраструктури ЦОД.

Метою розробки є дослідження та програмна реалізація системи кібербезпеки силової інфраструктури ЦОД.

Об'єктом дослідження є процес кібербезпеки силової інфраструктури ЦОД.

Предметом дослідження є методи кібербезпеки силової інфраструктури ЦОД.

Методи дослідження базуються на методах теорії захисту інформації, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи кібербезпеки силової інфраструктури ЦОД.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ архітектури IBM PC з ОС Windows XP/Vista/7/8/10.

Програму розроблено в середовищі Embarcadero RAD Studio Delphi 10.3.2 Rio Architect.

Ключові слова: комп'ютерна інженерія, кібербезпека, силова інфраструктура, ЦОД

ABSTRACT

Kapkan V.V. Research and software implementation of cybersecurity system of data center power infrastructure. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2021

In this final qualification work on the second (master's) level of higher education the software which is intended for system of cybersecurity of power infrastructure of the data center is developed.

The purpose of development is research and software implementation of the cybersecurity system of the power infrastructure of the data center.

The object of the study is the process of cybersecurity of the power infrastructure of the data center.

The subject of the research is the methods of cybersecurity of the power infrastructure of the data center.

Research methods are based on methods of information security theory, methods of mathematical statistics, methods of software development.

The result of the work is the software implementation of the cybersecurity system of the power infrastructure of the data center.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

Developed user-friendly interface. Instructions for working with software are given.

The program can be used on an IBM PC with Windows XP / Vista / 7/8/10.

The program is developed in Embarcadero RAD Studio Delphi 10.3.2 Rio Architect.

Keywords: computer engineering, cybersecurity, power infrastructure, data center

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ГБЗ	–	глобальна база знань
ДФ	–	дестабілізуючі фактори
ІС	–	інформаційна система
IoT	–	інтернет речей
IIoT	–	промисловий інтернет речей
ІТ	–	інформаційні технології
КФ	–	керуючі фактори
ЛОМ	–	локальна обчислювальна мережа
НКК	–	нечіткі когнітивні карти
ММЕ	–	міжмережний екран
ОЗП	–	оперативний запам’ятовуючий пристрій
ПЗ	–	програмне забезпечення
ПФ	–	проміжні фактори
ЦОД	–	центр обробки даних
ЦФ	–	цільові фактори
SSDT	–	таблиця дескрипторів системних сервісів

ВСТУП

Актуальність теми. Коли оператори центрів обробки даних розглядають питання кібербезпеки, вони звичайно думають про захист своєї IT-інфраструктури й розташовуваних у ЦОД даних. А коли вони думають про забезпечення безпеки своїх джерел електроживлення, вони думають про альтернативні джерела електрики, а також про обмеження фізичного доступу до своєї енергетичної інфраструктури.

Генератори, джерела безперебійного живлення й блоки розподілу живлення – все це допомагає забезпечувати й контролювати електроживлення центрів обробки даних. Але фахівці рідко приділяють досить увагу контролю кібербезпеки у своїх енергосистемах, хоча ці системи досить уразливі для кібератак. І, як це ні парадоксально, деякі із систем, використовуваних для захисту інфраструктури, можуть самі по собі являти загрозу безпеки.

Більшістю енергетичних устаткувань у центрі обробки даних можна управляти дистанційно й точно також налаштовувати його через віддалені термінали. Завдяки цьому зловмисник у теорії може одержати контроль над цими пристроями й перервати подачу електроживлення в центр обробки даних або на конкретний пристрій у мережі ЦОД.

Деякі із цих систем керування можуть потрапити в категорію інтернету речей (IoT), а якщо точніше – у сегмент промислового інтернету речей (IIoT). Пристрої класу IIoT є частиною невидимої інфраструктури центра обробки даних, що перебуває в «сірій зоні» між сферами відповідальності фахівців з керування фізичною інфраструктурою й фахівців з кібербезпеки.

Відповідно до доповіді організації Darktrace базованої в Сан-Франциско (США), кількість атак на IoT-пристрої зросло на 100 відсотків торік. А відповідно до опитування, проведеному торік організацією SANS Institute, тільки 40 відсотків компаній впроваджують патчи й фікси для захисту пристроїв IIoT.

При цьому цілих 56 відсотків респондентів заявили, що складності при впровадженні таких виправлень є однією із самих серйозних проблем безпеки для їхніх компаній. Крім того, майже 40 відсотків опитаних заявили, що в них виникли проблеми з пошуком відповідних пристроїв, відстеженням і керуванням ними.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи кібербезпеки силової інфраструктури ЦОД.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Огляд існуючих систем кібербезпеки силової інфраструктури ЦОД.

– Дослідження системи кібербезпеки силової інфраструктури ЦОД.

– Програмна реалізація системи кібербезпеки силової інфраструктури ЦОД.

Об'єктом дослідження є процес кібербезпеки силової інфраструктури ЦОД.

Предметом дослідження є методи кібербезпеки силової інфраструктури ЦОД.

Методи дослідження базуються на методах теорії захисту інформації, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод кібербезпеки силової інфраструктури ЦОД.

– Розроблено вітчизняний продукт кібербезпеки силової інфраструктури ЦОД, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі кібербезпеки силової інфраструктури ЦОД.

Достовірність наукових результатів підтверджена теоретичними викладеннями, даними комп'ютерного моделювання, коректними дослідженнями

параметрів на функціонуючій обчислювальній мережі, а також відповідністю отриманих результатів окремим результатам, наведеним у науковій літературі.

Робота апробована на LV Науково-технічна конференція здобувачів вищої освіти «Наука – виробництву», 2021, основні положення випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти надруковані у статті збірника праць молодих науковців ЦНТУ, випуск №12.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи кібербезпеки силової інфраструктури ЦОД, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Найбільш масштабні кібератаки останнього років були спрямовані на національну енергетичну інфраструктуру. Як приклад можна привести атаки на українські електричні мережі, які мали місце в 2015 і 2016 р.

Атаки на промислові середовища стали мейнстримом. В 2020 році кілька держав зафіксували триваючі атаки на свої енергетичні мережі. В 2020 році очікується збільшення числа голосних кібератак на цю критично важливу інфраструктуру.

У минулому злочинці, як правило, використовували зламані пристрої IoT для створення ботнетів, але після зараження пристрою його також можна використовувати для ряду інших зловмисних цілей. І хакери всі хащі це розуміють.

У випадку атак більшості типів групи кібербезпеки можуть ізолювати трафік або навіть цілі скомпрометовані системи. Але контроль за промисловим IoT – це особливий випадок. У промислових системах керування ізоляція трафіку або систем рідко є прийнятним варіантом, а внесення виправлень у реальному часі й зовсім є нежиттєздатною опцією.

Якщо пристрої й комп'ютери, що управляють електроживленням корпоративної або комерційної серверної ферми, будуть скомпрометовані, їхнє відключення може привести до припинення подачі електроживлення у весь центр обробки даних у цілому.

Є ще одна причина приділяти захисту доступу до енергосистем підвищена увага. Зловмисники, які одержують контроль над джерелами електроживлення центра обробки даних, можуть відключити центр обробки даних, але вони також можуть викликати стрибок напруги, що руйнує устаткування.

Щось подібне відбулося, хоча й випадково, в 2017 році в дата-центрі British Airways. За повідомленнями, технічний фахівець відключив систему керування від джерела безперебійного живлення, що привело до відключення електроживлення в ЦОД, що тривало кілька мінут. Потім подача електрики була знову відновлена, викликавши стрибок напруги, що, у свою чергу, викликало фізичне ушкодження устаткування.

1.2 Область застосування

Заражені IoT-системи, що є частиною допоміжної інфраструктури ЦОД, можуть становити небезпеку для інших мереж дати-центра. Оскільки ці пристрої пропонують точку входу в мережі центра обробки даних, ними потрібно управляти й захищати з тією же старанністю, що й сервери.

Існує кілька підходів, які оператори центрів обробки даних можуть використовувати для захисту цих систем керування. Мікросегментація, наприклад, може блокувати весь трафік на пристрій, за винятком дозволеного трафіку.

У деяких випадках це означає, що в кожного пристрою буде своя логічна, а не фізична мережа. Існують також спеціалізовані рішення для контролю доступу до мережі для електромереж, які активно блокують несанкціонований трафік.

Коли оператор центра обробки даних здобуває нові пристрої з підтримкою IoT для комплектації ними допоміжної інфраструктури, безпека повинна бути частиною комплексної перевірки. Варто переконатися, що паролі змінені із заводських, системи можуть бути оновлені, а настроювання пристроїв IoT – прийняті в увагу.

Іноді вибору може не бути, і оператор центра обробки даних повинен використовувати те, що доступно. Якщо є ризики, пов'язані із пристроєм з підтримкою IoT для комплектації допоміжної інфраструктури варто враховувати

такі ризики й планувати всі заздалегідь. Для кожного ризику повинен бути створений інструмент контролю, що компенсує. Наприклад, якщо інтерфейс керування IoT не може використовувати шифрування, спробуйте зашифрувати трафік через використання тунельного шифрування.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи кібербезпеки силової інфраструктури ЦОД, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

Повідомленнями про погрозу кібератак на основі нових і не нових виявлених уразливостей інформаційних систем мало кого можна здивувати в сучасному світі.

28.09.2021. Дослідники безпеки виявили новий IoT-ботнет Torii, по функціональних можливостях переважаючий Mirai і його численні варіанти. Можливості ботнету містять у собі розкрадання даних і віддалене виконання команд. За словами дослідників з Avast, шкідливе ПЗ активно щонайменше із грудня 2017 року.

(Ботнет, botnet – мережа комп'ютерів, які інфіковані шкідливим ПЗ, що дозволяють зловмисникам здійснювати віддалений контроль за ними й, зокрема, здійснювати через них DDoS-атаки. Зверніть увагу: шкідливе ПЗ виявили через майже рік після початку його діяльності.)

04.10.2021. На підпільних торговельних площадках пропонуються логіни/паролі до облікових записів користувачів Facebook. Вартість інформації варіюється від $3 до $12, а оплата приймається у віртуальній валюті (в основному bitcoin і bitcoin cash). Відповідно до звіту британської компанії Money Guru, на підпільних площадках недорого продаються логіни/паролі не тільки до облікових записів в Facebook, але й в інших сервісах. Наприклад, за облікові дані до аккаунтів користувачів Reddit продавці просять приблизно $2, Twitter – $3, Instagram – $6, Pinterest – $8.2

18.10.2021. Два мільярди пристроїв уразливі перед погрозою Blueborne, відкритої рік назад. Фахівці відзначають, що погрози Blueborne особливо

небезпечні для промислових підприємств, а пристрою інтернету речей можуть стати ґрунтом для нового ландшафту подібних атак.

25.10.2021. У Мережі з'явився новий ботнет, що атакує слабозахищені пристрої зі сфери "Інтернету речей", а також SSH-сервери й системи на базі Linux з метою проведення подальших DDoS-атак.

13.11.2021. Творець IoT-ботнету поширює бекдор для маршрутизаторів ZTE, у якому захований ще один бекдор для злому тих, хто його використовує. Як повідомляють експерти NewSky Security, хакер під псевдонімом Scarface входить у топ-20 найбільш значимих фігур на кіберзлочиній сцені в сфері «Інтернету речей». У зараженні ботнетів-конкурентів бекдорами є певний зміст. Заразивши системи "скрипт-кідді" (малодосвідчених хакерів), такий великий гравець, як Scarface може одержати контроль над створеними ними невеликими ботнетами або зруйнувати їх для усунення конкуренції.[5]

(Бекдор, backdoor (чорний хід) – дефект алгоритму, що навмисно вбудовується в нього розроблювачем і дозволяє одержати несанкціонований доступ до даних або віддаленого керування комп'ютером. Зверніть увагу: зловмисники-одинака конкурують між собою. Питання – за що саме? Тільки чи за головні ролі на кіберзлочиной сцені?).

Це – лише п'ять повідомлень про уразливості і атаки. З'являються такі повідомлення практично щодня.

Статистика кібератак і уразливостей

У жовтні 2021 р. компанія Cisco оприлюднила результати дослідження з кібербезпеки серед компаній малого й середнього бізнесу, у якому взяли участь 1816 респондентів з 26 країн. За результатами дослідження з'ясувалося, що:

– Більше 53% невеликих підприємств в 2021 р. Піддавалися кібератакам, а 20% з них заявили про збиток у розмірі від 1 до 2,5 млн. долл.

– 53% респондентів заявили, що їхні компанії піддавалися вторгненням, що спричинило істотні фінансові витрати.

– В 40% респондентів (підприємства із чисельністю 250-499 співробітників) в 2021 році в результаті серйозних атак траплялися 8-годинні простої.

– В 39% респондентів половина систем постраждала в результаті тої або іншої серйозної атаки.

Страждають від кібератак не тільки дрібні компанії, але й великі, і навіть деякі країни. Наприклад, APT-група BlackEnergy протягом от уже декількох років атакує Україну. Зокрема, під час першого у світі відключення електроенергії, що відбулись в результаті кібератаки (у грудні 2015 р.) без електрики залишилися 230 тис. чоловік.

Компанія ESET повідомляє про виявлення спадкоємця APT-групи BlackEnergy. Група зловмисників, що одержала назву GreyEnergy, націлена на шпигунство й розвідку й, цілком можливо, готується до майбутніх атак з метою кіберсаботажу. «GreyEnergy протягом останніх трьох років бере участь в атаках на енергетичні компанії й інші мети особливої важливості в Україні й Польщі», – повідомляють фахівці ESET.

Про те, наскільки серйозними можуть бути погрози й наслідки атак, говорить абревіатура APT (advanced persistent threat – розвинена стійка погроза), що означає, з одного боку, цільову кібератаку, з іншого боку – групу людей, що володіють спеціальними знаннями й значними ресурсами. Метою атаки може бути розкрадання інформації, створення перешкод діяльності або навіть припинення роботи структури, що стала жертвою нападу. Ще один варіант – створення умов для нападу в майбутньому, коли збиток може стати максимальним.

Постраждати від кібератак можуть і звичайні користувачі.

19.10.2021. Від атаки мережного шахра постраждало 10 тис. чоловік. Кримінальний бізнес зловмисника із псевдонімом Investimer, що займається шахрайством на ринку криптовалют, відрізняє різноманітні асортименти

використовуваного шкідливого ПЗ й багатий набір способів незаконного заробітку.[8]

22.10.2021. Check Point у звіті Global Threat Index за вересень відзначає, що кількість атак майнерів криптовалюти на пристрої Apple iPhone збільшилося майже на 400%. Атаки проводяться за допомогою шкідливого ПЗ Coinhive, що займає верхній рядок у рейтингу Global Threat Index із грудня 2017 р.

Дослідники Check Point також проаналізували найбільш експлуатованої уразливості. Перше місце зберегла уразливість CVE-2017-7269, що торкнулася 48% організацій по усьому світі. На другому місці – проблема CVE-2017-5638 із глобальним охватом в 43%, на третьому, з невеликим відставанням, – можливість ін'єкції коду через невірну конфігурацію PHPMyAdmin на веб-сервері. Ця уразливість виявлена в 42% компаній.

Отже, деякі найбільш експлуатованої уразливості виявлені майже в половини організацій по усьому світі. Якщо якась компанія не має однієї уразливості, вона цілком може мати іншу, або навіть трохи. І тоді кібератака на неї – лише справа часу.

Саме тому деякі фахівці з кібербезпеки розшифровують абревіатуру «Інтернет речей» (IoT, Internet of Things) як Internet of Threats ("Інтернет погроз").

Кібербезпека IoT

Кількість уразливостей для персон і компаній, що використовують інтернет речей (IoT) і тим більше промисловий інтернет речей (IIoT), набагато більше, ніж для інших компаній, а також для громадян, що не проживають в «розумних» будинках і підключень, що використовують мінімальну кількість, до Інтернету. Багато в чому це обумовлено тим, що збільшується кількість пристроїв, через які зловмисники можуть провести кібератаку. Зокрема, це датчики, пристрої граничних (мрячних) обчислень, виконавчі пристрої й ін.

З огляду на важливість цих проблем, Агентство ЄС по мережній і інформаційній безпеці (European Union Agency For Network And Information

Security) у листопаді 2021 р. випустило звіт про велике дослідження «Вихідні рекомендації з безпеки для IoT у контексті критичних інформаційних інфраструктур». (Baseline Security Recommendations for IoT in the context of Critical Information Infrastructures).

У ході дослідження експертам було запропоновано, зокрема, оцінити 10 сценаріїв атаки з погляду критичності. На основі даних, зібраних з інтерв'ю експертів, була оцінена середня критичність сценаріїв атаки. Для різних сценаріїв вийшли такі оцінки:

– Атака на систему керування IoT – 88,89%.

– Атака на датчики, зміну зчитувальних ними значень або граничних значень і настроювань – 84,72%.

– Атака на пристрої, уведення команд керування – 81,94%.

– Атака на ланку мережі між контролерами й виконавчими пристроями – 73,61%.

– Атака на інформацію, передану через мережу – 73,61%.

– Атака на виконавчі пристрої, зміну або скидання їхніх нормальних значень – 69,44%.

– Використання уразливостей протоколу – 62,50%.

– Stepping stones attack (атака із проміжних хостів, які були заздалегідь скомпрометовані) – 58,33%.

– Маніпуляція джерелом живлення – 58,33%.

Крім того, експерти оцінили як найнебезпечніші наступні три сценарії атаки:

– Сценарій 1: компрометація системи керування IoT.

– Сценарій 2: маніпуляція значеннями пристроїв IoT.

– Сценарій 3: введення ботнетів/команд.

Сценарій 3, наприклад, має на увазі використання деякої уразливості усередині пристрою для уведення через нього команд і одержання прав адміністратора з метою створення ботнету, що складає з даного й інших

уразливих пристроїв IoT. Такий сценарій атаки може бути реалізований за допомогою ботнету Mirai, згаданого на початку статті. Цей ботнет провів трохи найдужчих DDoS-атак у новітній історії й довів свою здатність атакувати різні об'єкти: від присвяченого кібербезпеки веб-сайту KrebsOnSecurity до телекомунікаційної інфраструктури всієї країни. Тому вплив атаки Mirai на інфраструктури з небезпечними енергетичними ресурсами може досягати надзвичайно критичних рівнів.

IIoT: проблем з безпекою набагато більше

Цитоване вище дослідження присвячене інтернету речей (IoT), нас же більше цікавить промисловий інтернет речей (IIoT). Наскільки застосовні до нього наведені вище оцінки й сценарії?

Деякі експерти думають, що єдине розходження між названими концепціями – це їхньої області використання. У той час як IoT найбільше часто застосовується для споживчих цілей, IIoT використовується в промисловості – у сфері виробництва, для керування ланцюжками поставок і в системах керування.[11]

Однак реальність трохи складніше. Пристрій IoT може мати таку ж функціональність, що й пристрій IIoT, і все-таки не вважатися промисловим продуктом. До пристроїв, мережам і системам керування IIoT пред'являються значно більше тверді й набагато більше великі вимоги, чим до аналогічних компонентів IoT. І насамперед це відноситься до безпеки. Адже порушення процесу великого виробництва може привести до того, що вартість невипущеної через цього продукції складе мільйони доларів у день. Порушення роботи електромережі впливає на економічну активність мільйонів людей і відносить під погрозу національну безпеку. Тому в рішеннях IIoT використовуються істотно розширені міри безпеки – захищеної й стійкі системні архітектури, спеціалізовані набори мікросхем, шифрування й автентифікація, системи виявлення погроз і т.д.

Крім того, до IIoT пред'являються такі вимоги, як:

– Сумісність із численними вже наявними на підприємстві системами виробництва й керування.

– Придатність для масштабування при росту обсягу виробництва.

– Висока точність всіх вимірів і реакцій на події.

– Здатність до частого перепрограмування, гнучкість і адаптуємість.

– Висока швидкодія, мінімальні затримки у виявленні відхилень, їхній оцінці, прийнятті рішень і реагуванні.

– Висока надійність у тяжких умовах експлуатації.

– Стійкість до відмов окремих пристроїв і підсистем.

– Придатність до підвищення рівня автоматизації, аж до повного виключення втручання персоналу в робочий процес.

– Зручність обслуговування й ремонту, у ході яких може вироблятися заміна датчиків, відновлення убудованого програмного забезпечення, настроювання шлюзів і серверів.

Очевидно, при виконанні деяких з перерахованих вище вимог виникають додаткові погрози, нехарактерні для IoT. Наприклад, у ході обслуговування й ремонту може вироблятися заміна датчиків, відновлення убудованого ПЗ, настроювання шлюзів і серверів. Датчики – другий по ступені небезпеки компонент. Нове ПЗ може мати нові, ще не виявлені фахівцями з кібербезпеки уразливості. Через помилки, зроблених у ході настроювання шлюзів і серверів, можуть виникнути додаткові уразливості. Вони можуть виникнути також через часте перепрограмування, виконуваного, наприклад, при кожній зміні або модифікації продукції, що стає усе більше різноманітної, а іноді взагалі робиться під індивідуальні замовлення. Сумісність із уже наявними на підприємстві системами виробництва й керування означає, у тому числі, взаємодія із застарілими програмними продуктами, при розробці яких ураховувалися набагато менш тверді вимоги по безпеці. І це – лише мала частина додаткових погроз, характерних для IIoT.

Заходу щодо запобігання атак

У цитованому вище дослідженні «Вихідні рекомендації з безпеки для IoT...» представлений докладний перелік мер безпеки й передових практик, спрямованих на зм'якшення погроз, уразливостей і ризиків, характерних для пристроїв і середовища IoT.

Основні міри безпеки IoT можна розділити на три основні категорії:

– Політики. Перший набір мер безпеки відноситься до політиків, спрямованим на забезпечення інформаційної безпеки й на те, щоб зробити її більше конкретної й надійної. Вони повинні відповідати діяльності організації й містити добре документовану інформацію. Засоби забезпечення безпеки повинні передбачатися ще в процесі розробки пристроїв/додатків IoT, впроваджуватися при їхньому виробництві й розгортанні й підтримуватися протягом усього життєвого циклу всієї системи IoT на всіх її рівнях.

– Організаційні, технологічні міри й робота з персоналом. Всі підприємства повинні мати організаційні критерії інформаційної безпеки. Їхня кадрова політика повинна сприяти забезпеченню безпечного керування процесами й інформацією. Організації повинні домогтися того, щоб їхні підрядники й постачальники також відповідали за інформаційну безпеку в зонах своєї компетентності. Організація повинна бути підготовлена до можливих інцидентів, пов'язаним з безпекою (зона відповідальності, оцінка й реакція).

– Технічні міри. Очевидно, що для зменшення уразливості IoT міри безпеки й практика їхнього застосування повинні ставитися до всіх технічних пристроїв. При цьому варто враховувати особливості екосистеми IoT, зокрема такі, як масштабованість. З урахуванням величезної кількості задіяних в IoT пристроїв, у забезпечення деяких мер безпеки може знадобитися введення в архітектуру екосистеми спеціалізованих компонентів, наприклад, шлюзів.

Всі ці міри й кращі практики їхнього застосування дуже докладно розглянуті в цитованому вище дослідженні.

Слід зазначити, що проблеми кібербезпеки не можуть бути вирішені раз і назавжди. Для підтримки ефективності системи безпеки її потрібно не тільки безупинно контролювати, але й регулярно обновляти. Наприклад, зловмисники, ідучи в ногу з технічним прогресом, почали активно застосовувати технології машинного навчання, тому сучасна платформа кібербезпеки вже не може обійтися без використання штучного інтелекту.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Embarcadero RAD Studio Delphi 10.3.2 Rio Architect – це найшвидший спосіб створювати й обновляти інтенсивно працюючі з даними, сильно взаємодіючі застосунки з візуально насиченим користувальницьким інтерфейсом для Windows 10, Mac, мобільних пристроїв, IoT і інших платформ за допомогою Object Pascal і C++. Широкий вибір функцій підтримки Windows 10, у тому числі нові компоненти VCL для Windows 10, стилі для VCL і FMX, а також служби UWP (універсальної платформи Windows), наприклад повідомлення, дозволяють легко й швидко перенести застосунки в Windows 10, зберігши користувачів. Нова платформа дозволяє підтримувати великі проекти на більшому числі платформ із подвоєним обсягом пам'яті в середовищі розробки й удвічі більшим розміром підтримуваних проектів. Крім того, підтримка декількох моніторів і десятки нових функцій середовища розробки, призначених для прискорення створення коду, зроблять роботу як ніколи ефективною. За допомогою RAD Studio 10 розроблювачі зможуть створювати застосунки в 5 разів швидше в порівнянні з іншими інструментами, а розробка застосунків для декількох настільних, мобільних, хмарних платформ і платформ баз даних, включаючи 32і 64 -розрядні версії Windows 10, Mac OS X, iOS і Android, стане ще швидше.

Зміни у версії 10.3 Rio:

– Створюйте міжплатформені застосунки. 80% інтернет користувачів мають смартфони й застосунки доступу, а також дані з мобільного пристрою й ноутбука / настільного комп'ютера, саме тому так важливо в цей час, щоб застосунки працювали в будь-якому пристрої.

– У всіх версіях Professional, Enterprise і Architect RAD Studio 10.3 надається підтримка процесу розробки застосунків для мобільних пристроїв. Розроблювачі RAD Studio кодують лише один раз, компілюють споконвічно для кожної платформи, що скорочує час і трудозатрати на вивчення декількох мов і дозволяє паралельно управляти циклами розробки.

– Підтримка Android API26, відповідність вимогам Google Play Store відносно нових застосунків із серпня 2018 року й відновлення застосунків з листопада 2018 року.

– Власні елементи керування Android і стилізовані елементи керування FMX в одній і тій же формі Android, включаючи тему матеріального дизайну для Android 5.0 або вище.

– Підтримка iOS 12 (32і 64-біт) для створення App Store і корпоративних застосунків.

– Підтримка смайликів Юнікод.

– Програмуйте по-своєму. Завдяки двом новим темам самостійне налаштування інтегрованого середовища розробки для відповідності вашому стилю кодування ще ніколи не була настільки простій.

– Темне й світле оформлення Незалежно від того чи волієте ви кодувати вночі або у світлий час доби, завдяки темному й світлому оформленню RAD Studio ви можете вибрати потрібний вам стиль. Було доведено, що темне оформлення допомагає знизити зорову напругу в умовах низького освітлення, дозволяючи вам працювати більш продуктивно вночі. Немає нічого простіше, ніж перейти від темного до світлого оформлення й навпаки за допомогою меню панелі інструментів.

– Виконаєте користувальницьке налаштування свого середовища розробки Поліпшена програма установки інтерфейсу користувача й менеджера ліцензій інтерфейсу користувача дозволяє визначити ті можливості, які необхідні й опустити непотрібні, незалежно від того чи розробляєте ви застосунки для декількох платформ або всього однієї.

– Чистий, оновлений інтерфейс користувача інтегрованого середовища розробки Знайдіть потрібні можливості. Швидко. Головне вікно інтегрованого середовища розробки відцентровано й відрізняється високим ступенем читаності. Ви з легкістю визначите, де перебуває область фокусування клавіатури з оновленими змінами фонових квітів фокуса. Вкладки редактора більше, що полегшує читання шрифтів, тому ви можете швидко внести зміни й зберегти кодування.

– Чудові застосунки Windows з VCL. Бібліотека візуальних компонентів (Visual Component Library, VCL) пропонує просту й візуальну розробку користувальницького інтерфейсу застосунки, у версії 10.3 представлені нові відновлення, які дозволять вашим додаткам виглядати сучасними й свіжими.

– Розширена підтримка HighDPI. Завдяки новому елементу керування VCL High DPI ImageList у версії 10.3 розроблювачі, що створюють нові застосунки VCL для Windo ws або оновлюючі існуючі застосунки для High DPI дисплеїв, можуть повністю підтримувати зроблені до рівня пікселів зображення зі змінною розв’язною здатністю на всіх елементах керування, а також будь-яке користувальницьке креслення, що вимагає масштабованих зображень для моніторів з різною розв’язною здатністю.

– Підтримка Per Monitor V2. Переконаєтеся, що ваш додаток масштабується правильно для всіх типів масштабування в Windows, реагуючи на зміни масштабування DPI на різних екранах під час виконання.

– Розширена підтримка Windows 10 і WinRT API. Сюди відноситься ряд ключових API-інтерфейсів WinRT і останні API-інтерфейси Windows 10,

включаючи готові до використання компонента для убудованих у застосунки покупок і випробувань у магазині Windows 10 Store.

– Розгортання застосунків на основі служб за допомогою RAD Server. Продуктивність RAD Server була значно поліпшена завдяки десятикратному збільшенню потужності відносно простих операцій.

– Нові компоненти обробки JSON допоміжного засобу.

– Розширена підтримка RAD Server для клієнта Ext JS. Об'єднайте зовнішній інтерфейс javascript і веб-службу, підтримувану REST Server REST. (У версії Architect тепер включена ліцензія ExtJS Professional).

– Версії Enterprise включають ліцензію для одиничного розгортання RAD Server.

– Версії Architect включають ліцензію для розподіленого розгортання RAD Server.

– Що нового в C++? Підтримка C++17 Win32 збільшує продуктивність, поліпшує роботу компілятора й прискорює процес кодування. Були оновлені RTL і STL.

– Нова версія STL/Dinkumware 2018 для Win32 і Win64.

– Поліпшене автодоповнення коду Автодоповнення коду для даного компілятора тепер асинхронне, швидше й із кращими результатами, чим у попередньому автодоповненні коду C++. Уведення тексту не буде припинятися, поки виконується розрахунок.

– Тепер є підтримка налагодження для оптимізації компонувань.

– 2X швидкість математичної продуктивності для Win64.

– Нові додаткові лабораторії C++ в GetIt.

– Нові й поліпшені можливості роботи з базами даних. InterBase 2017 / IBToGo 2017 в RAD Studio. Версії Professional включають ліцензію розроблювача InterBase 2017, у той час як версії Enterprise і Architect містять у собі ліцензії InterBase ToGo. InterBase ToGo доповнена можливістю шифрування, функціями

зміни подань, призначених для простої синхронізації даних застосунку по підписці без обмежень за розміром файлу бази даних.

– Поліпшена й оновлена підтримка для популярних баз даних, включаючи mySQL v8.0, MariaDB 10.3, SQL Server 2017, PostgreSQL v10, Firebird v3.0, MongoDB, InterBase, SQLite 3.23.1, SQL Anywhere і багатьох інших.

– Удосконалення DataSnap.

– Поліпшення REST. Підтримка додаткових родинних REST методів, типів і властивостей.

– Повністю оновлений модуль живлення версії Architect. Одержіть більше від версії Architect, включаючи ці ліцензії сімейства Idera.

– Ліцензія Sencha ExtJS Professional: Створіть свій ідеальний мережний вхідний інтерфейс за допомогою javascript і ExtJS.

– Ліцензія на розгортання InterBase ToGo. Додайте сховище даних у свої застосунки за допомогою цієї гнучкої, зашифрованої бази даних, що вбудовується.

– Ліцензія для розподіленого розгортання RAD Server. Ідеально підходить для серверного застосунку архітектури мікросервісів.

– Ліцензія AquaData Studio. Вражаючий аналіз бази даних.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи кібербезпеки силової інфраструктури ЦОД.

В процесі розробки випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Інтерес до різних IoT-пристроїв з боку зловмисників продовжує рости: за першу половину 2020 року ми одержали в три рази більше зразків шкідливого ПЗ, що атакує «розумні» пристрою, чим за весь 2021 рік. До слова, в 2021 їх було в 10 разів більше, ніж в 2018 році. Як бачите, тренд «чим далі, тим гірше» простежується дуже добре.

Ми вирішили вивчити, які вектори атак використовуються зловмисниками для зараження «розумних» пристроїв, які зловреди завантажуються в систему в результаті успішної атаки, а також чим все це може обернутися для власника пристрою й просто жертв нового ботнету.

Одним із самих популярних векторів атак і, відповідно, зараження пристроїв усе ще залишається перебір пароля Telnet. У другому кварталі 2020 року таких атак на ханипоти було в 3 рази більше, ніж всіх інших разом узятих.

Найчастіше зловмисники завантажували на IoT-пристрої один зі зловредів сімейства Mirai (20,9%).

Отже, у другому кварталі 2020 року лідером по кількості унікальних IP-адрес, з яких виходили атаки через перебір пароля Telnet стала Бразилія (23%). Друге місце, з невеликим відривом, зайняв Китай (17%). Росія ж у нашім списку зайняла 4-ое місце (7%). Усього за період з 1 січня по липень 2020 року на наших Telnet-ханипотах було зафіксовано більше 12 мільйонів атак з 86 560 унікальних IP-адрес, а шкідливе ПЗ завантажувались із 27 693 унікальних IP-адрес.

Оскільки частина власників «розумних» пристроїв змінює штатний пароль Telnet на більше складний, а багато хто гаджети й зовсім не підтримують цей протокол, зловмисники перебувають у постійному пошуку нових шляхів зараження. Цьому сприяє й висока конкуренція між вірусописьменниками, яка

привела до зниження ефективності атак з перебором пароля: у випадку успішного злому пристрою його пароль міняється, а доступ до Telnet блокується.

Як приклад використання «альтернативної техніки» можна привести ботнет Reaper, у його активі за станом на кінець 2017 року налічувалося близько 2 мільйонів IoT-пристроїв. Замість перебору паролів до Telnet цей ботнет експлуатував відомі уразливості в ПЗ:

– Уразливості в прошиванні роутера D-Link 850L.

– Уразливості в IP-камерах GoAhead.

– Уразливості в CCTV камерах MVPower.

– Уразливість в Netgear ReadyNAS Surveillance.

– Уразливість в Vacron NVR.

– Уразливість у пристроях Netgear DGN.

– Уразливості в роутерах Linksys E1500/E2500.

– Уразливості в роутерах D-Link DIR-600 і DIR 300 – HW rev B1.

– Уразливості в пристроях AVTech.

Переваги такого способу поширення в порівнянні з перебором пароля:

– Зараження відбувається значно швидше.

– Закрити уразливість у ПЗ значно складніше, ніж перемінити пароль або відключити/заблокувати сервіс.

Незважаючи на те, що цей спосіб складніше в реалізації, він прийшовся до смаку багатьом вірусописьменикам і нові троянци, що використовують відомі уразливості в ПЗ «розумних» пристроїв, не змусили себе довго чекати.

Нові атаки – старі зловреди

Щоб подивитися, які уразливості намагаються експлуатувати зловреди, ми проаналізували дані про спроби підключень до різних портів на наших пастках.

Поки переважна більшість атак – це усе ще перебір паролів Telnet і SSH. На третім місці по популярності перебувають атаки на сервіс SMB, що надає віддалений доступ до файлів. Ми поки не бачили IoT-зловредів, що атакують цей сервіс. Однак деякі його версії містять серйозні відомі уразливості – EternalBlue

(Windows) і EternalRed (Linux) -за допомогою яких, наприклад, поширювався сумно відомий троянець-шифрувальник WannaCry і шкідливий майнер криптовалюти Monero EternalMiner.

Як можна помітити, з більшим відривом лідирують пристрою компанії MikroTik, що працюють під керуванням RouterOS. Причина, як видно, в уразливості Chimay-Red. [1]

Порт 7547

Досить популярні атаки на сервіс віддаленого адміністрування пристроїв (специфікація TR-069), що працює на порту 7547. По даним Shodan, у світі налічується більше 40 мільйонів пристроїв, у яких цей порт відкритий. І це незважаючи на те, що ще не дуже давно уразливість стала причиною зараження мільйона роутерів Deutsche Telekom, а також «допомогла» поширенню зловредів сімейств Mirai і Hajime.

Ще один тип атак експлуатує уразливість Chimay-Red у роутерах MikroTik під керуванням RouterOS версії нижче 6.38.4. У березні 2020 року з її допомогою активно поширювався Hajime.

IP-камери

Зловмисники не обійшли стороною й IP-камери: у березні 2017 року в ПЗ пристроїв GoAhead було виявлено трохи серйозних уразливостей, а через місяць після публікації інформації про це з'явилися нові модифікації троянців Gafgyt і Persirai, що експлуатують ці уразливості. Уже через тиждень після початку активного поширення цих зловредів число заражених пристроїв виросло до 57 тисяч.

8 травня 2020 року був опублікований proof-of-concept до уразливості CVE-2020-10088 веб-сервера XionMai uc-httpd, що використовується в деяких «розумних» пристроях китайських виробників (наприклад, відеореєстратор KKMoon DVR). Уже наступного дня зареєстрована кількість спроб виявити пристрої, що використовують даний веб-сервер, виросла більш ніж в 3 рази.

Винуватцем такого сплеску активності став троянець Satori, що раніше атакував роутери GPON.

Нові зловреди й погроза кінцевому користувачеві

DDoS-атаки

Головним завданням, що зловмисники вирішують за допомогою IoT-зловредів, була й залишається організація DDoS-атак. Заражені «розумні» пристрої стають частиною ботнету, що по команді починає атакувати зазначену адресу, позбавляючи хост можливості нормально обробляти запити реальних користувачів. Такими атаками, наприклад, продовжують займатися троянци сімейства Mirai і його клони, зокрема, зловред Hajime.

Це, мабуть, найбільш необразливий сценарій для кінцевого користувача. Максимум, що загрожує власникові зараженого пристрою (і це дуже малоймовірний сценарій)- блокування з боку інтернет-провайдеру. А «вилікувати» пристрій найчастіше можна за допомогою простого перезавантаження.

Видобуток криптовалюти

Інший тип корисного навантаження пов'язаний із криптовалютами. Наприклад, IoT-зловреди можуть установлювати майнер на заражений пристрій. Але через невелику обчислювальну потужність «розумних» пристроїв, доцільність такої атаки залишається під сумнівом, навіть незважаючи на їх потенційно велику кількість.

Більше хитрий і діючий спосіб збагатитися на парочку-іншу криптомонеток придумали автори троянця Satori. IoT-пристрій виступає в їхньому сценарії в ролі своєрідного «ключа», що відкриває доступ до високопродуктивного ПК:

– На першому етапі зловмисники намагаються заразити якнайбільше роутерів, використовуючи відомі уразливості, а саме:.

– CVE-2014-8361 – RCE в miniigd SOAP service в Realtek SDK;

– CVE 2017-17215 – RCE у прошиванні роутерів Huawei HG532;

– CVE-2020-10561, CVE-2020-10562 – обхід авторизації й можливість виконати довільні команди на роутерах Dasan GPON.

– CVE-2020-10088 – переповнення буфера в XiongMai uc-httpd 1.0.0, що використовується в прошиваннях деяких роутерів і інших «розумних» пристроїв деяких китайських виробників;

– Використовуючи скомпрометовані роутери й уразливість CVE-2020-1000049 у механізмі віддаленого адміністрування ПЗ для майнинга криптовалюти Etherium – Claymore, замінити адреса гаманця на свій;

Крадіжка даних

Виявлений у травні 2020 року троянець VPNFilter переслідує інші мети. Головна серед них – перехоплення трафіку зараженого пристрою, добування з нього важливих даних (логіни, паролі й т.п.) і відправлення їх на сервер зловмисників. От основні особливості VPNFilter:

– Модульна архітектура. Автори зловреда можуть «дооснащать» його новими функціями «на лету». Так, на початку червня 2020 р. був виявлений новий модуль, що вмів инжектить javascript-код у перехоплені веб-сторінки;

– Стійкість до перезавантажень пристрою. Троянець прописує себе в стандартний для Linux-систем планувальник crontab, а також може змінювати конфігураційні параметри в енергонезалежній пам'яті (NVRAM) пристрою;

– Використання TOR для комунікації зі своїм сервером керування;

– Можливість самознищення й виводу пристрою з ладу. Одержавши відповідну команду, троянець видаляє себе, а також перезаписує сміттєвими даними критичну частину прошивання після чого перезавантажує пристрій;

Спосіб поширення троянця дотепер залишається невідомим: його код не містить ніяких механізмів самопоширення. Однак ми схильні думати, що для зараження використовуються відомі уразливості в ПЗ пристроїв.

У найпершому звіті про VPNFilter говорилося про 500 тисячі заражених пристроїв. З тих пор їх стало більше, а список виробників уразливих гаджетів значно збільшився. На середину червня він включав наступні бренди:

– ASUS.
– D-Link.
– Huawei.
– Linksys.
– MikroTik.
– Netgear.
– QNAP.
– TP-Link.
– Ubiquiti.
– Upvel.
– ZTE.

Збільшує ситуацію ще й те, що пристрою цих виробників використовуються не тільки в корпоративних мережах, але часто й у якості домашніх роутерів звичайних користувачів.

«Розумних» пристроїв стає усе більше й, по деяких прогнозах, до 2020 року їхня кількість у кілька разів перевищить населення планети. Однак виробники усе ще приділяють недостатньо увагу їхньої безпеки: немає нагадувань про необхідність зміни стандартних паролів при першому настроюванні, немає повідомлень про вихід нових версій прошивань, а сам процес відновлення може бути складний для звичайного користувача. Все це робить IoT-пристрою підходящою метою для зловмисників. Їх простіше заразити ніж персональний комп'ютер і при цьому вони можуть займати далеко не останнє місце в домашній інфраструктурі: одні управляють усім інтернет-трафіком, інші можуть знімати відео, а треті управляють іншими пристроями (наприклад, кліматичною установкою).

Шкідливе ПЗ для «розумних» пристроїв розвивається не тільки кількісно, але і якісно: в арсеналі зловмисників з'являється усе більше експлойтів, використовуваних для самопоширення, а заражені пристрої використовуються

для крадіжки персональних даних і майнинга криптовалют, а не тільки для організації DDoS-атак.

От кілька простих рад, які допоможуть мінімізувати ризик зараження «розумних» пристроїв:

– Не відкривайте доступ із зовнішньої мережі до пристрою без гострої потреби;

– Періодичне перезавантаження допоможе позбудеться від уже встановлених зловредів (але в більшості випадків ризик повторного зараження залишається);

– Регулярно перевіряйте наявність нових версій прошивання й обновляйте пристрій;

– Використовуйте складні паролі довжиною не менш 8 символів, що включають у себе букви різного регістра, цифри й спецсимволи;

– Міняйте заводські паролі після першого запуску пристрою, при первісному настроюванні (навіть якщо пристрій про це не просить);

– Закрийте/заблокуйте «зайві» порти, якщо є така можливість. Наприклад, якщо ви не підключаєтеся до роутеру по Telnet (порт tcp:23), варто відключити його, щоб перекрити зловмисникам можливу лазівку;

3.2 Розробка структурної схеми

Як було сказано вище більшістю енергетичних устаткувань у центрі обробки даних можна управляти дистанційно й точно також налаштовувати його через віддалені термінали. Ці енергетичні устаткування містять у собі мікроконтролери. Безпека системи, побудованої на базі мікроконтролера – це системний захист убудованого програмного забезпечення й даних, розглянутих як найважливіша ланка загальної функціональності системи. Збиток, що може нанести несанкціонований доступ до паролів і персональної інформації, досить очевидний, але в захисті бідує також і сама програма пристрою. Доступ до

фрагментів коду, навіть у вигляді бінарних файлів, залежно від намірів зловмисника може привести до обходу ліцензійних і програмних обмежень, копіюванню ключових алгоритмів і навіть до клонування всього пристрою й створенню репліки. Ця репліка, завдяки відсутності витрат на розробку програмного забезпечення, може бути істотно дешевше оригіналу, що дозволить видалити з ринку первинного розроблювача.

Результатом виконання вимог безпеки завжди є ускладнення розроблювальної системи. Додаткові витрати на апаратні рішення й додатковий час, витрачений на розробку й тестування коду, відбиваються на вартості й складності підтримки кінцевого пристрою.

Особливо чутливими ці витрати стали тепер, у процесі масового впровадження IoT, коли, з одного боку, пристрою на мікроконтролерах повинні ставати усе більше масовими й усе більше дешевими, з іншого боку – робота в мережах IoT, де постійно відбуваються підключення до виділеного сервера або хмарного сервісу, висуває підвищені вимоги до інформаційної безпеки системи. До того ж майстерність атакуючої сторони росте, як і скоординованість дій окремих хакерів і реверс-інженерів. Порию доходи, отримані навіть від одиничних успішних атак, покривають видатки на численні невдалі спроби, а з урахуванням масовості ринку IoT, ці доходи мають тенденцію до росту.

Сімейство STM32G0 стало відповіддю фірми STMicroelectronics на найчастіше суперечливі вимоги до мікроконтролерів для ринку IoT. Вдало об'єднавши невисоку ціну, енергоефективність і розширений арсенал убудованих апаратних інструментів, відповідальних за безпеку, STM32G0 на базі ядра ARM Cortex-M0+ може стати основою системи, що не тільки задовольнить зростаючі запити до продуктивності й економії енергії, але й буде максимально захищеної без надмірних складностей у розробці й супроводі.

Щоб знати свого ворога, розглянемо можливі типи атак, потім поговоримо про загальні способи організації протидії, і наприкінці зосередимося на конкретних методах захисту, пропонованих сімейством STM32G0.

Атаки

При розгляді питань безпеки завжди необхідно пам'ятати про ключову аксіому миру інформаційної безпеки – навіть незважаючи на повноцінний комплекс початих захисних заходів, атака все-таки може увінчатися успіхом.

По-перше, які б міри не вживали для комплексного захисту системи, цілком імовірно, що нова, раніше невідома, пролом у безпеці буде виявлена й використана протягом терміну служби пристрою. Існує ціла екосистема, що поєднує програмістів, хакерів і фахівців з реверс-інжинірингу, де свіжознайдена «Уразливість нульового дня» (англ. «0day»), тобто неусунута помилка або шкідлива програма, проти якої ще не розроблений захисний механізм, служить не тільки предметом гордості, але й прибутковим товаром. Із цього треба простий вивід – у плині всього строку експлуатації приладу необхідно вживати певні зусилля для дослідження нових типів атак і для періодичного відновлення ПЗ.

По-друге, нижче ми розглянемо, наскільки високотехнологичними й дорогими можуть бути деякі техніки доступу до вмісту мікроконтролера. З погляду атакуючого, необхідно максимізувати співвідношення «очікувана вигода»/«витрати на атаку». Очікувана вигода пропорційна цінності добутої інформації, вартість атаки пропорційна ціні використовуваного устаткування й часу, витраченого на злом. З обліком цього можна сказати, що вжиті захисні заходи не роблять злом зовсім неможливим, але збільшують вартість атаки, роблячи злом справою усе менш вигідним. Зрозуміло, при збільшенні цінності інформації, закладеної в мікроконтролер, для зниження співвідношення «вигода»/«витрати» рекомендується розглянути можливість застосування більше складних (і, найчастіше, більше дорогих) методів захисту. І навпаки, якщо цінність умісту мікроконтролера невелике, можливо пропорційне зниження витрат на забезпечення безпеки.

При оцінці сумарної вартості злому необхідно також мати на увазі повторюваність атаки, тобто можливість повторного злому тим же самим набором інструментів, що знову приводить нас до виводу про необхідність періодично піддавати ревізії й модифікувати методи захисту від атак.

Типи атак

Розглянемо можливі види атак на мікроконтролер, починаючи з базових, і аж до професійно спланованих атак із застосуванням спеціалізованого інженерного устаткування. Існують наступні типи атак: атака програмного забезпечення (ПЗ), неінвазивна атака апаратного забезпечення й інвазивна атака апаратного забезпечення; також варто окремо розглянути специфіку атаки IoT-системи.

У таблиці 3.1 наведений короткий огляд розглянутих атакуючий технік.

Таблиця 3.1 – Типи атак

Тип атаки	Атака ПЗ	Апаратна неінвазивна	Апаратна інвазивна
Умови	Місцево або віддалено	Потрібен доступ до зібраного пристрою або до друкованої плати	Потрібен доступ до друкованої плати
Методики	Помилки в ПЗ Уразливості інтерфейсів Сканування інтерфейсів Фаззинг	Відлагоджувальний порт Комунікаційні порти Впроваджений код (code injection) Перешкоди по ланцюгах живлення (power glitches) Позаштатне тактирування (clock glitches) Аналіз побічних ефектів (side-channel attack)	Зондування кристала Лазерне різання Травлення Іонне травлення (FIB) Оптична й електронна мікроскопія Інжекція перешкод (fault injection)

Продовження таблиці 3.1

Вартість	Від дуже низької до високої, залежно від рівня безпеки, наміченого при розробці убудованого ПЗ	Від середньої до високої. Потрібен обмежений набір щодо нескладного устаткування й персонал, що володіє середньою кваліфікацією	Дуже висока. Необхідно дороге спеціалізоване устаткування й персонал, що володіє експертною кваліфікацією
Мети	Доступ до конфіденційної інформації (код і дані), перехоплення даних, відмова устаткування	Доступ до конфіденційної інформації й визначення алгоритмів роботи пристрою	Реверс-інжиніринг (одержання повного пакета документації для відтворення пристрою)

Атака програмного забезпечення

Використовуються уразливості властиво ПЗ, уразливості протоколів обміну й перехоплення каналів зв'язку. У переважній більшості випадків використовуються саме програмні атаки, тому що вони характеризуються низькою вартістю й високим нанесеним збитком. Набір устаткування для програмної атаки досить дешевий (найчастіше можна обійтися персональним комп'ютером з мінімальною кількістю розповсюджених інтерфейсних пристроїв), а співтовариство хакерів і реверс-інженерів, що практикують програмні зломи, дуже добре злагоджено.

Помилки в ПЗ дозволяють перевести пристрій у позаштатний режим роботи за допомогою таких програмних процедур, як переповнення буфера або ініціація витоків пам'яті. У цьому випадку атакуюча сторона може одержати, залежно від типу помилки, доступ до закритих даних або навіть повний контроль над системою.

Використання **уразливостей інтерфейсів** припускає доскональне знання деталей використання конкретних протоколів на базі конкретної апаратної платформи, що дає можливість робити атаки, засновані на тонкощах реалізації, таких, як робота з буфером, особливості використання програмної купи й стека, деталі функціонування стандартних бібліотек. **Сканування інтерфейсів** припускає запис переданих даних у вигляді, придатному для подальшого аналізу.

Одним зі спеціалізованих видів атаки ПЗ є **фаззинг** (fuzzing), коли на інтерфейси й канали уведення інформації приладу подаються свідомо неправильні, випадкові дані або дані в обсязі, що сильно перевищує штатні обсяги. Зловмисники зацікавлені як у виявленні закономірності зависань і помилкової роботи, здатних прояснити особливості внутрішньої логіки пристрою, так і в ініціації процесу витоку пам'яті, здатного в результаті дати доступ до коду, який захищається.

Неінвазивні методи атак апаратного забезпечення

При неінвазивной атаці захист обходиться за допомогою підключення до виводів мікроконтролера й інших використовуваних у приладі мікросхем.

Доступ до незаблокованого **відлагоджувальному порту** дозволяє атакуючий уважати дампи Flash-пам'яті, перетворити які в добре вихідний код, що досить читається, мовою високого рівня, наприклад, на Си – тільки справа часу. Крім того, запускаючи процес функціонування приладу, що зламується, під контролем відлагоджувального порту, зловмисник одержує доступ не тільки до властиво алгоритму, але й до тонкостей його взаємодії з конкретним оточенням.

Доступ до **комунікаційних портів** дозволяє застосувати спеціалізовані сканери інтерфейсів або записати процедури обміну інформацією для їхнього подальшого аналізу. Цей вид атаки в цілому схожий на сканування інтерфейсів з розділу програмних атак, але більше дійственен, тому що в атакуючої сторони з'являється доступ до внутрішніх комунікаційних ресурсів аналізованого пристрою – до I^2C, SPI, 1-Wire, паралельним шинам зовнішньої пам'яті й іншим.

У випадку, якщо система не захищена від запуску стороннього коду, використовується так званий впроваджений код (code injection) – програма, що запускається на зламується оборудовании, що, і виконуючій функції сканування всіх доступних областей пам'яті для подальшого одержання бінарного прошивання пристрою, або активуюча процедури, здатні змусити пристрій змінити функціонування відповідно до мет атакуючої сторони – записувати дані, отримані по комунікаційних інтерфейсах, зберігати паролі доступу, передавати інформацію стороннім користувачам.

Перешкоди по ланцюгах живлення (power glitches) і **позаштатне тактировані** (clock glitches) можуть перевести мікроконтролер у некоректний режим роботи, здатний дати атакуючій стороні додаткову інформацію про режими роботи пристрою. **Аналіз побічних ефектів** (side-channel attack) припускає сканування й аналіз додаткових параметрів, що характеризують роботу мікроконтролера, таких, як споживаний струм, температура корпуса, електромагнітна емісія. Ця інформація, сама по собі досить незначна, укупі з іншими добутими даними може дати ключ до розумію внутрішньої структури програми.

Інвазивні методи атак апаратного забезпечення

Інвазивна атака передбачає руйнуючий вплив, що забезпечує зловмисникові прямий фізичний доступ до кристала мікроконтролера.

До найпростішого способу інвазивної атаки варто віднести **зондування кристала**, коли за допомогою пошарового шліфування або **лазерного різання** розкривається корпус мікросхеми й на провідні зв'язки усередині кристала, які можуть служити контактними площадками, установлюються мікрозонди, що з'єднують досліджувану схему із зовнішнім устаткуванням, наприклад, з логічними аналізаторами або аналізаторами протоколів.

За допомогою шліфування, різання, **травлення** й **оптичної** або **електронної мікроскопії** можна встановити пошарову структуру кристала, що при відповідних навичках можна використовувати для одержання не тільки функціональної, але й принципової схеми чипа.

Інжекція перешкод використовується аналогічно перешкодам по ланцюгах живлення й позаштатному тактуванню при неінвазивних апаратних атаках, але тільки на новому, більше тонкому й продуктивному рівні.

Нарешті, **іонне травлення** за допомогою іонної гармати (апарата, принципово схожого на електронний мікроскоп, але потік, що використовує, важких іонів) дозволяє робити на мікрорівні дуже широкий спектр філігранних маніпуляцій, таких, як різання провідників, напилювання нових струмопровідних доріжок і навіть створення на існуючій підложці нових транзисторів. Іонна гармата – дуже тонкий і потужний інструмент, нерідко використовуваний навіть розроблювачами мікросхем для корекції пробних кристалів.

Як неінвазивні, так і інвазивні апаратні атаки вимагають фізичного доступу до зламується устройству, що. Найпоширеніший, очевидний і дешевий метод полягає у використанні незахищеного відлагоджувального порту, але, найчастіше атака апаратного забезпечення виливається в досить нетривіальний, складний і дорогий захід, виконуваний із залученням спеціалізованого устаткування, специфічних матеріалів і висококваліфікованих фахівців.

Методи атак IoT-систем

Ключова особливість IoT-пристрою полягає в тому, що крім мікроконтролера, сенсорів і виконавчих механізмів, воно в обов'язковому порядку містить у своєму составі канал зв'язку з інтернет-сервером, що залежно від завдань, поставлених перед системою, може зберігати й обробляти дані, забезпечувати взаємодія з користувачем і відповідати за інтеграцію й взаємодію всіх пристроїв, підключених до даної мережі. Крім того, інтернет-сервер відповідальний за контроль працездатності й відновлення програмного забезпечення підключених до нього IoT-пристроїв.

З погляду безпеки можна виділити наступні актуальні вектори атак на IoT-пристрій:

– широка номенклатура (Ethernet, Wi-Fi, Bluetooth, LoRa і т.д.) і досить велика кількість уразливостей каналів зв'язку. Є виражена тенденція до

збільшення кількості пристроїв, підключених через бездротові інтерфейси й різновиди PLC (Power line communication, використання ліній електропередачі для обміну даними), що не вимагають для підключення й атаки фізичного контакту із пристроєм;

– тісний взаємозв'язок IoT-пристрою й сервера визначає можливість злому не пристрою, а саме сервера, з наступним доступом до каналу зв'язку й пристрою. Завдання ця, з одного боку, досить складні, тому що серверна частина розподілених IoT-систем часто базуються на хмарних технологіях (Amazon Web Services, Google Cloud, Microsoft Azure), захист яких з боку провайдеру перебуває на дуже високому рівні. З іншого боку, треба мати на увазі можливість використання власних серверів, не завжди належним чином захищених, поширеність методів злому серверного устаткування й зкоординованість співтовариства хакерів;

– IoT-пристрій, що працює, як правило, у мережі собі подібних, повинне коштувати недорого, інакше сумарна вартість рішення може виявитися непід'ємною. З огляду на ефект масштабу, розроблювач у процесі проектування повинен заощаджувати буквально кожний цент, у тому числі зважуючи раціональність застосованих мір, спрямованих на забезпечення безпеки, і іноді мимоволі приймаючи рішення, що тяжіють до економії, а не до безпеки. І отут досить до речі буде згадати про головних героїв нашої статті, мікроконтролерах серії STM32G0. Як буде показано нижче, STMicroelectronics при розробці цієї серії зробило упор саме на економії й безпеці, так що тепер їсти можливість сконструювати максимально захищене, але при цьому бюджетний пристрій;

Автори досліджень, пов'язаних з безпекою IoT (наприклад, «Privacy and the Internet of Things» Гілада Роснера), також відзначають появу таких системних ефектів, як зосередження на серверах величезних обсягів різнохарактерної інформації, несанкціоноване використання якої здатно привести до великого збитку, і змушений допуск до роботи персоналу, що найчастіше не володіє належною кваліфікацією й недостатньо стурбованого інформаційною безпекою.

Методи протидії

Після розбору видів можливих атак ми досить обґрунтовано можемо спрогнозувати й піддати аналізу методи захисту (рисунок 3.1).

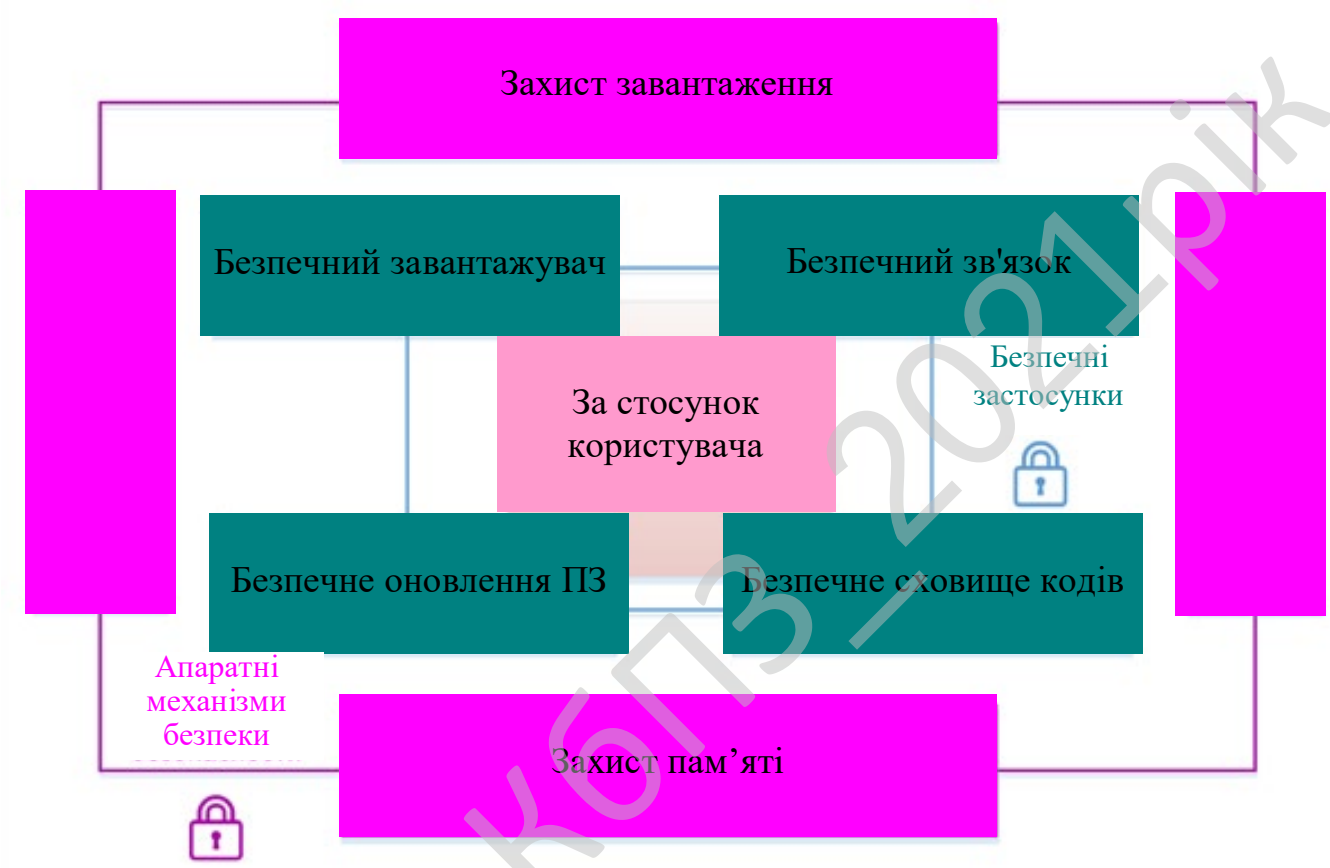

Рисунок 3.1 – Структурна схема системи

Захист Flash-пам'яті

Без сумніву, найважливіший і найдійовіший метод захисту – блокування доступу до пам'яті пристрою. Утримуюча пропрієтарний код і чутлива дані, Flash-пам'ять мікроконтролера в обов'язковому порядку повинна бути закрита як для зовнішніх інтерфейсів (таких як відлагоджувальні порти), так і від внутрішніх неавторизованих процесів.

У цілому, при плануванні атаки на мікроконтролер саме вміст Flash є метою номер один. Завантажник, основна програма й необхідні для роботи дані в Flash-пам'яті, разом зі схемою приладу, отриманої при аналізі апаратного

забезпечення, сукупно здатні скласти повний пакет документації, необхідний для клонування пристрою.

При роботі з лінійкою STM32G0 для комплексного захисту пам'яті від **зовнішньої атаки** необхідно в першу чергу активувати **RDP** (тут і далі курсивом виділені конкретні захисні механізми, наявні в арсеналі STM32G0 і детально розібрані в наступній главі). Саме RDP відключає можливість зчитувати Flash-пам'ять через відлагоджувальний інтерфейс JTAG або SWD. Це перший рубіж оборони й свого роду вододіл, що змушує зловмисника переходити від простого копіювання інформації до осмислених атакуючих заходів.

При здійсненні **внутрішньої атаки** на Flash-пам'ять зловмисник намагається запустити сторонній код, інжектований в ОЗП або попередньо впроваджений у сторонню бібліотеку, для того, щоб одержати доступ до основних коду й даним. Метод непростий, але дуже продуктивний, захиститися від зовнішнього коду, що зчитує вміст внутрішньої пам'яті, досить важко, але, на щастя, у серії STM32G0 передбачений цілий арсенал від такого виду атак:**PCROP**, що забороняє доступ на читання, запис і навіть на стирання до певного блоку Flash-пам'яті; **Securable memory area**, що блокує доступ до завантажника після його однократного виконання; **MPU**, що дозволяє гнучко налаштовувати права доступу до Flash-пам'яті, ОЗП і регістрам.

Захист від запису невикористовуваної пам'яті допоможе запобігти інжекції коду або модифікацію основної програми. STMicroelectronics рекомендує не залишати невикористану пам'ять у вихідному стані (virgin value), а заповнювати її яким-небудь передбачуваним паттерном, наприклад, командою NOP, а краще кодом виклику переривання або свідомо некоректними значеннями (illegal op-codes), при спробі виконання яких буде також викликане переривання; у такому випадку мікроконтролер одержить інформацію про те, що в результаті збоїти або навмисного втручання почалося виконання коду з невикористовуваної області пам'яті й потрібно вжити відповідних заходів. Для захисту від запису, крім згаданого вище MPU, варто використовувати спеціалізований інструмент -

WRP, за допомогою якого можна заблокувати на запис два незв'язаних блоки Flash-пам'яті різного розміру.

Документація від STMicroelectronics відзначає, що в список захисних заходів варто обов'язково включити активацію механізму **ECC**, що дозволяє автоматично детектувати і виправляти помилки пам'яті. ECC не є чистим засобом захисту від атак, будучи, відповідно до «AN4750, Handling of soft errors in STM32 applications», у першу чергу методом виявлення випадкових помилок (random failures control techniques), але при цьому входить в обов'язковий чек-лист активуємих захистів.

Захист ОЗП

Стік, масиви даних, буфери комунікаційних інтерфейсів і змінні – все це розташовано в ОЗП і є бажаною метою для хакера.

Досить поширене **виконання коду з ОЗП**, тому що саме ОЗП – найшвидша пам'ять, наявна в розпорядженні мікроконтролера, і найчастіше ділянка коду, при виконанні якого потрібна максимальна швидкість, переносять перед виконанням в ОЗП. Іншою причиною для виконання коду з ОЗП є наявність зовнішньої Flash-пам'яті, що містить зашифрований код, що переноситься в оперативну пам'ять, де згодом розшифровується й виконується. Зрозуміло, у такому випадку потрібно захистити ОЗП (або його частина) від зовнішнього доступу. У цьому нам допоможе MPU, що здатний гнучко настроїти права доступу до пам'яті або навіть включити режим, у якому виконання коду з оперативної пам'яті повністю заборонене (execute never).

Після виконання деяких ділянок коду ОЗП може містити тимчасові значення секретних змінних. Наприклад, при відправленні або прийомі зашифрованого повідомлення пароль, що перебуває в захищеному секторі Flash-пам'яті, може бути зчитаний в ОЗП. Потрібно в обов'язковому порядку **очищати ОЗП** відразу після виконання операцій, здатних залишити у вільному доступі секретні дані. Всі тимчасові змінні повинні бути віддалені, буфери комунікаційних протоколів обнулені, стек і купа повинні бути зачищені. На

додаток до цього рекомендується використовувати RDP для заборони завантаження з ОЗП й заборони завантаження із зовнішніх джерел (bootload).

Разом з ECC, детектуючим помилки Flash-пам'яті, працює **контроль парності ОЗП**, що автоматично виявляє помилки в ОЗП. Контроль парності рекомендується до обов'язкової активації, навіть незважаючи на те, що без нього обсяг оперативної пам'яті виростає на 12.5 % (1/8, додається по одному біті на кожний байт).

Ізоляція програмного забезпечення

Ізоляція коду відноситься до високорівневих технік захисту від злому, і припускає спільну скоординовану взаємодію апаратних можливостей, передбачених виготовлювачем мікроконтролера, розроблювачами операційних систем і програмістами, що пишуть прикладний код. Отут треба в черговий раз пом'янути добрим словом інженерів STMicroelectronics, що включили Memory protection unit до складу навіть самих недорогих лінійок STM32G0 Value Line, і тим самим заложивших потужний фундамент безпеки для всього сімейства.

Ізоляція ПЗ базується на можливості відокремлення окремих процесів, які не мають змоги впливати один на одного. Кожний процес має свою дозволену область роботи, власний стік даних, і не може вплинути на інші процеси. Ізоляцією процесів друг від друга управляє операційна система, і ця ізоляція переслідує двояку мету:

– захистити одні процеси від помилок, можливих в інших процесах. Обвалення одного із процесів через переповнення або витоки пам'яті може бути коректно оброблене й не приведе до втрати працездатності всього приладу;

– захистити одні процеси від навмисного шпигунського втручання з боку іншого процесу. Конфіденційні дані, такі, як криптографічний ключ або секретний алгоритм, будуть доступні в рамках тільки одного процесу, що виконується;

Для коректної ізоляції ПЗ разом з MPU рекомендується використовувати Securable memory area, за допомогою якої можна визначити деякі параметри процесів ще на рівні завантажника, що не модифікується.

Відлагоджувальні порти й інтерфейси завантаження

Обов'язкове відключення відлагоджувальних портів, реалізоване за допомогою RDP, уже було згадано в розділі, присвяченому захисті Flash-пам'яті, і є зовсім обов'язковою, першочерговою захисною мірою.

Крім цього, потрібно в обов'язковому порядку відключати невикористовувані можливості по завантаженню ззовні. Наприклад, навіть найпростіший мікроконтролер серії STM32G0, STM32G070, може завантажити зовнішній код з USART1, USART2, USART3, I2C1, I2C2, SPI1, SPI2 (глава «STM32G07xxx/08xxx device bootloader» в «AN2606. STM32 microcontroller system memory boot mode»). Це величезна діра в безпеці, і, якщо завантаження із зовнішнього джерела не передбачена конструкцією приладу, її потрібно обов'язково відключити за допомогою RDP.

Моніторинг стану системи

Програма мікроконтролера повинна постійно відслідковувати показання наявних на борті датчиків, для того, щоб при виході їхніх показань на межі припустимого, вживати заходів по зміні режиму роботи на більше безпечний. Деякі показники стану системи, призначені в першу чергу для захисту від випадкових збоїв, можуть, проте, застосовуватися для детектування атаки, що почалася на пристрій. Наприклад, спадання напруги живлення нижче припустимої границі або відключення зовнішнього тактового сигналу можуть означати як ненавмисну апаратну проблему, так і хакерську атаку.

Механізм забезпечення безпеки тактування **CSS** забезпечує безпечне відключення зовнішніх тактових частот HSE і LSE у випадку їхньої відмови або ненормального поводження й перехід на роботу від убудованих генераторів.

Моніторинг напруги живлення за допомогою убудованого АЦП (для мікроконтролерів лінійки Value Line) і програмувальний детектор напруги **PVD** (для лінійок Access Line і Access Line & Encryption) можуть сигналізувати про вихід напруги живлення за межі норми й про необхідність переходу в безпечний режим.

Вимір температури кристала за допомогою убудованого сенсора дозволить визначити перегрів або занадто сильне охолодження приладу.

Виявлення фізичного доступу (**Tamper detection**) – функція, відповідальна винятково за безпеку. З її допомогою можна визначити розкриття корпуса приладу й ужити заходів для зміни режиму робота або знищення чутливої інформації.

Безпечне завантаження й відновлення ПЗ

Після скидання мікроконтролера завантаження різних компонентів програми відбувається поетапно: на самому початку мікроконтролер налаштовує опції забезпечення безпеки системи (MPU, IWDG і т.д.), потім завантажується користувальницький код, і наприкінці – сторонні бібліотеки. Забезпечення безпеки системи протягом цього покрокового процесу, коли більше довірений елемент перед завантаженням перевіряє безпека менш довіреного, називається «ланцюжком довіри» (chain of trust, рисунок 4) і забезпечується наступними процедурами:

– забезпеченням максимальної захищеності кореневого компонента (root). Якщо кореневий компонент буде скомпрометований, то буде скомпрометований весь ланцюжок безпеки, і всі інш завантажувати^ся компоненти, що, будуть під загрозою;

– перевіркою контрольних сум установлюваних компонентів (за допомогою MD5 або SHA256);

– максимальним використанням у процесі завантаження криптографічних схем;

Якщо при завантаженні всі операції можна фізично ізолювати усередині корпуса мікроконтролера, то при відновленні ПЗ канал доставки **завжди** вважається небезпечним. Навіть якщо ви обновляєте програму мікроконтролера через захищений Ethernet або передаючи прошивання за допомогою GSM-Модему, з метою безпеки краще розглядати відновлення, як, скажемо, бінарний

файл, переданий користувачеві на пристрої USB Flash, і споконвічно припускати, що зацікавлена в зломі сторона обов'язково одержить доступ до цієї інформації.

Виходячи з таких передумов, видно, що відновлення ПЗ мікроконтролера **завжди** повинне бути криптографічно захищено, і, отже, мікроконтролер завжди повинен бути здатний автентифікувати, перевірити цілісність і встановити таке криптографічно захищене відновлення.

3.3 Розробка функціональної схеми

Функціональна схема розробленої системи зображена на рисунку 3.2.

З рисунку видно, що розроблена система складається з наступних частин:

– Блок виявлення дестабілізуючого трафіку у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT.

– Блок визначення деструктивних дій та виду атаки у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT.

– Блок визначення деструктивних дій та виду атаки у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT.

– Блок зберігання результатів.

– Блок моніторингу мережі у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT.

– Блок аналізу мережної статистики у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT.

– Блок визначення топології мережі у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT.

Блок виявлення дестабілізуючого трафіку у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT

Блок виявлення дестабілізуючого трафіку у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT:

– Блок визначення профілю поводження нормального трафіку у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT.

– Блок заміни направлення трафіку у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT.

– Блок усунення дестабілізуючого трафіку у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT.

При первісному розгортанні рішення по DDoS адміністратор створює профіль поводження нормального трафіку у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT. Цей процес іменується навчанням. Компанія використовує додатки звичайним образом протягом 24 годин протягом одного тижня, і трафік додатка проходить через Детектор аномалій трафіку у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT. У період навчання Детектор аномалій трафіку у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT збирає базову інформацію для розуміння нормальної роботи мережі у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT, куди входять:

– Інтенсивність пакетів для кожного типу пакетів, обмірювана як кількість пакетів у секунду (pps).

– Співвідношення пакетів, наприклад, співвідношення пакетів SYN і пакетів FIN.

– Кількість одночасних TCP-з'єднань, відкритих одним джерелом.

Базова інформація збирається по кожній цільовій адресі хост-пК, цільовій підмережі у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT, вихідній адресі хост-пК і вихідній підмережі у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT.

Після закінчення періоду навчання Детектор аномалій трафіку у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT переводиться в режим моніторингу, а Блок усунення дестабілізуючого трафіку у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT – у резервний режим готовності. Доти, поки немає атаки у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT, що активно розвивається, вхідний трафік з мережі у системі кібербезпеки силової

інфраструктури ЦОД побудованої на пристроях IoT Інтернет проходить через комутатор без якого-небудь втручання з боку Блоку усунення дестабілізуючого трафіку у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT. Копія вхідного трафіку у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT посилає для аналізу на Детектор аномалій трафіку у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT через зовнішній аналізатор протоколів (SPAN) або віртуальні списки ACL. Якщо Детектор аномалій трафіку у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT виявляє дестабілізуюче в порівнянні з базовою інформацією поводження трафіку у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT, починається процес усунення:

– Детектор аномалій трафіку у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT направляє в Блок усунення дестабілізуючого трафіку у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT команду почати процес зміни напрямку.

– Блок усунення дестабілізуючого трафіку у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT відхиляє (“захоплює”) трафік, адресований на атакуєму IP-адресу, переадресуючи його на самого себе.

– Блок усунення дестабілізуючого трафіку у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT піддає трафік багатоступінчастому аналізу й застосовує контрзаходи для відділення благонадійних джерел від джерел атаки у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT. Цей процес іменується очищенням або вичищенням.

– Блок усунення дестабілізуючого трафіку у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT скидає трафік атаки у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT й пересилає благонадійний трафік назад на нормальний маршрут проходження трафіку у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT до мети. Цей процес іменується ін'єкцією.

Детектор аномалій трафіку у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT

Детектор аномалій трафіку у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT – це пасивний пристрій моніторингу, що постійно виявляє ознаки, що вказують на присутність атаки у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT DDoS, спрямованої проти захищеного місця призначення, також іменованого зоною. Це може бути сервер, інтерфейс міжмережного екрана або інтерфейс маршрутизатора. Детектор аномалій трафіку у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT аналізує копії всього вхідного трафіку у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT, адресуємого в захищені зони, через SPAN або відгалуження пасивної мережі у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT. Цей аналіз включає зіставлення поточного поводження трафіку у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT з базовими граничними параметрами, які також іменуються зональною політикою, для виявлення дестабілізуючого трафіку у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT. Якщо дестабілізуюче поводження виявлене й виглядає як можлива атака, Детектор аномалій трафіку у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT через позаполосну управлінську мережу Ethernet посилає в Блок усунення дестабілізуючого трафіку у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT сигнал про початок аналізу й усунення атаки у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT.

Блок усунення дестабілізуючого трафіку у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT

Блок усунення дестабілізуючого трафіку у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT – це автономний пристрій аналізу й фільтрації трафіку у системі кібербезпеки силової інфраструктури ЦОД

побудованої на пристроях IoT. Починаючи прийом трафіку у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT, адресованого в конкретну зону, що, очевидно, піддається атаці, Блок усунення дестабілізуючого трафіку у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT проводить точний аналіз цього трафіку у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT. Якщо результати аналізу підтверджують, що трафік злочинний, Блок усунення дестабілізуючого трафіку у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT застосовує контрзаходи, наприклад, механізми анти-спуфінга й фільтрацію різного рівня. Кінцевий результат полягає в тому, що трафік зі злочинних джерел скидається, а трафік із благонадійних джерел пересилається в передбачений пункт призначення.

Атаки у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT DDoS – Виявлення й усунення

У таблиці 3.3 перераховані типи атак DDoS, які може виявляти й усувати Блок усунення дестабілізуючого трафіку у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT.

Можливі варіанти зміни напрямку трафіку у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT

Фахівці з IT можуть використовувати описані нижче варіанти зміни напрямку трафіку у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT з його пересиланням з мережі у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT, розташованого вище лежачого оператора зв'язку, на Блок усунення дестабілізуючого трафіку у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT. Цей процес також іменується “захватом” трафіку у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT:

– Повідомлення прикордонного шлюзового протоколу (Border Gateway Protocol, BGP) із Блок усунення дестабілізуючого трафіку у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT на маршрутизатори, розташовані у вище лежачого оператора зв'язку, з інформацією про те, що трафік, адресований на захищену адресу призначення, буде переспрямований на Блок усунення дестабілізуючого трафіку у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT.

– Використання зовнішніх механізмів зміни напрямку трафіку у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT, наприклад, маршрутизаторів віддаленого відновлення BGP.

– Повідомлення про ін'єкцію очищеного трафіку у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT на маршруті (Route Health Injection, RHI) від Блок усунення дестабілізуючого трафіку у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT для процесу маршрутизації в Catalyst серії 6500 або в систему нагляду серії 7600. Ці повідомлення поміщають статичний маршрут у глобальну таблицю маршрутизації, у якій модуль Блок усунення дестабілізуючого трафіку у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT позначений як наступний вузол.

Можливі варіанти ін'єкції трафіку у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT

Ін'єкція трафіку у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT – це процес, застосовуваний у Блок усунення дестабілізуючого трафіку у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT для пересилання очищеного благонадійного трафіку у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT в точку призначення, що піддається атаці. Рішення підтримує різні варіанти ін'єкції трафіку у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT. У варіанті 2 -ого рівня топології, очищений трафік

пересилається із Блок усунення дестабілізуючого трафіку у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT на статично-конфігуруєму наступну адресу заходу. Ця адреса перебуває на маршрутизаторі, розташованому нижче й з'єднаним з тої ж VLAN або підмережею, що й інтерфейс/VLAN ін'єкції трафіку у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT. Ін'єкцію трафіку у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT на 2-му рівні найпростіше конфігурувати, оскільки тут не потрібно вносити які-небудь істотні зміни в конфігурацію маршрутизатора, розташованого нижче.

Варіанти ін'єкції трафіку у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT 3-го рівня:

– Маршрутизація й пересилання по VPN (VPN Routing and Forwarding, VRF).

– Маршрутизація на основі політики (Policy-Based Routing, PBR).

– Транкинг VLAN (VLAN Trunking).

– Інкапсуляція по загальній маршрутизації (GRE) або інкапсуляція IP у тунелі IP (IPIP).

Блок визначення деструктивних дій та виду атаки у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT

Експерти по захисту даних уже давно рекомендують IoT-виробникам дотримуватися принципів наскрізної інформаційної безпеки. Відповідно до них, безпека IoT-продукту повинна забезпечуватися на всіх етапах життєдіяльності девайса: від його створення до утилізації.

Але найчастіше виробники послуг і пристроїв у сфері IoT не приділяють цьому питанню належної уваги, використовуючи незахищену хмарну інфраструктуру або неперевірене ПЗ.

У зв'язку із цим виникає ряд проблем:

– відсутня підтримка з боку виробників, навіть коли уразливість знайдена;

– важко або неможливо обновити ПЗ й ОС;

– уразливість одного гаджета дозволяє проникнути у всю мережу;

Всі ці фактори приводять до частішання хакерських атак на IoT-системи.

Сценарії атак на кібербезпеку IoT-систем

У кожного сценарію атаки є свій рівень критичності – низький, середній або високий. Усього у звіті виділено 12 зон ризику, з них ми розглянемо шість найнебезпечніших.

1. Атаки на датчики IoT-пристроїв

Несправний датчик може пропустити стрибок потужності, що приведе до фізичного ушкодження системи. Наслідку цієї атаки – маніпуляції апаратним і програмним забезпеченням, відмова або несправність датчика / виконавчого механізму, збій або несправність системи керування.

2. Атаки на мережні дані

Ця атака спрямована на дані, які передаються по Мережі. На рівні контролера й системи керування (DCS, SCADA) дії хакерів непомітні, тому що власники системи бачать правильні значення. Маніпуляції можна виявити тільки за допомогою моніторингу трафіку мережного рівня.

Атака може вплинути на виробничий процес або завдати серйозної шкоди підприємству. Приміром, зміни температури печі, які не вдалося визначити заздалегідь, можуть спровокувати вибух пристрою.

3. Атака на шлюзи IIoT

Цей тип атаки складається з декількох фаз і звичайно запускається потай. Зловмисник намагається зламати шлюз IIoT, потенційно ставлячи під погрозу все інформаційне середовище. Атака особливо небезпечна, якщо на підприємстві використовують слабкі паролі або протоколи за замовчуванням.

У цьому випадку зловмисник одержує доступ до даних, пристроям, системам і мережному устаткуванню. У ході цієї атаки можуть украсти паролі, персональні дані, запустити шкідливе ПЗ й DDoS.

4. Атаки на пристрої дистанційного керування

Зловмисники також можуть зламати пристрою, які перебувають далеко від системи керування, наприклад смартфони. Що носяться гаджети легко зламати, тому через них хакери можуть одержати доступ до систем, де на це треба було б більше часу.

Парольні атаки, використання уразливостей програмного забезпечення, перехоплення сеансів, розкриття інформації, шантаж – все це може стати наслідком атаки на пристрої дистанційного керування.

5. Атаки, які відбуваються за допомогою людських помилок

Атаки цього типу найчастіше «обрушуються» на більші корпорації, де доступ до систем адміністрування є в багатьох людей. Діри в системі безпеки, спровоковані людським фактором, складно виявити через нетехнічний характер. Це може бути помилкове використання пристроїв, ненавмисна зміна даних у системі ОТ або фізичне ушкодження устаткування.

Атака може привести до збою системи або стати частиною більше складних маніпуляцій, пов'язаних із крадіжкою даних, грошей або інтелектуальної власності.

6. Атаки з використанням штучного інтелекту (AI)

Такі атаки вимагають більших тимчасових і грошових витрат, тому найчастіше виробляються на масштабні системи, наприклад системи промислового Інтернету речей. Завдяки штучному інтелекту зловмисники можуть поєднувати потенційні дані, отримані з Інтернету, з відомими даними, прискоривши пошук діри в безпеці.

Ці атаки найчастіше націлюються на конкретних людей, наприклад системних адміністраторів. Можуть стати причиною втрати даних при розвідці Мережі.

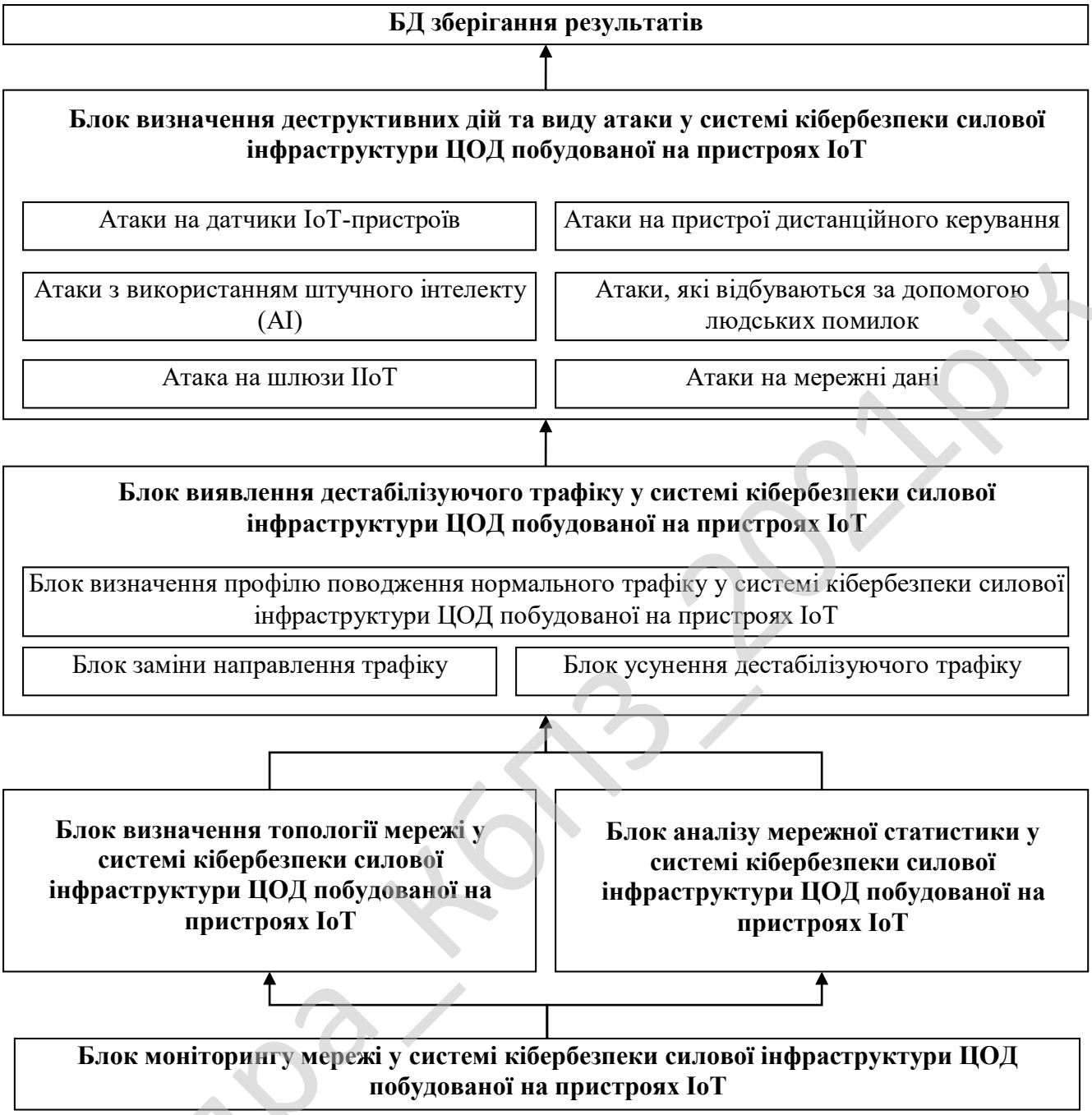

Рисунок 3.2 – Функціональна схема системи

Організаційні практики по захисту IoT-систем

Виробники IoT-пристроїв повинні убезпечити свої продукти ще на етапі їхнього створення. У цьому більшу роль грає організація роботи самої компанії, що займається розробками в сфері Інтернету речей.

Тому агентство сформулювало організаційні принципи для виробників IoT-пристроїв:

– забезпечити безпека програмного й апаратного забезпечення на кожному етапі життєвого циклу продукту;

– ураховувати міри безпеки по всьому ланцюжку поставок;

– проводити «польові випробування» кібербезпеки, щоб визначити відповідність її принципів технічної специфікації продуктів;

– налагодити безпечне ведення документації на всіх стадіях проекту: від створення IoT-пристрою або послуги до її експлуатації;

– створити цілісну архітектуру безпеки IoT-системи;

– контролювати дотримання вимог у встановленій архітектурі безпеки;

– виявляти й оперативно розслідувати кожна незвичайна подія, пов'язане з безпекою;

– установити процес обробки інцидентів (ідентифікації порушених активів, виявлення й класифікація уразливостей, їхнє усунення);

– розглянути питання про створення операційного центра кібербезпеки (SOC), де цими питаннями будуть займатися фахівці;

Технічні практики по захисту IoT-систем

Крім організаційних моментів, безпека також залежить від технічних можливостей IoT-середовища. Міри технічної безпеки реалізуються безпосередньо в самих девайсах:

– перевіряти надійність програмного забезпечення перед його запуском;

– авторизувати всі пристрої IIoT у системі OT, використовуючи відповідні методи, наприклад цифрові сертифікати або PKI;

– сформувати список каналів обміну даними між пристроями IIoT і вибрати самі безпечні;

– створити «білі» списки додатків і переглядати список не рідше одного разу в рік;

– обмежити доступ до систем шляхом детальної авторизації;

– надавати вхід у систему тільки обмеженому колу осіб з найменшими привілеями доступу;

– створити пристрій, у якому привілейований код і процеси ізольовані від частини прошивання, що не бідує у взаємодії з ними;

– впровадити DDoS-стійку й збалансовану інфраструктуру;

– переконатися, що веб-інтерфейси повністю шифрують сеанс користувача, від пристрою до серверних служб, і що вони не піддані атакам XSS, CSRF, SQL-ін'єкціям і т.і.

Захист IoT-систем від несанкціонованого доступу набирає популярність. За даними аналітичної компанії Gartner, світові витрати на кіберзахист IoT-систем з 2018 по 2021 рік виросли на 29% – з $912 млн до $1,17 млрд. Поява додаткових інвестицій, практичних рекомендацій і фахівців з кібербезпеки може позитивно вплинути на розвиток ринку Інтернету речей у цілому.

Блок аналізу мережної статистики у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT

Блок збирання наступної інформації:

– Основна статистика (Summary).

– Ієрархія протоколу (Protocol Hierachy).

– Сеанси обміну пакетами (Conversations).

– Точки призначення (Endpoints).

– Графіки I/O (IO Graphs).

– Список сеансів обміну пакетами (Conversation List).

– Список точок призначення (Endpoint List).

– Час чекання відповіді від сервісу (Service Response Time).

– RTP.

– SIP.

– Виклики VoIP (VoIP Calls).

– Призначення (Destination).

– Графік потоку (Flow Graph).

– HTTP.

– IP-адреса (IP address).

– Довжина пакету (Packet Length).

– Тип порту (Port Type).

Розпишемо їх більш детально.

1. Основна статистика. Доступні такі елементи основної статистики у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT, як:

– Властивості захоплених файлів.

– Час захвату.

– Інформація про фільтр захвату.

– Інформація про фільтр відображення.

2. Ієрархія протоколу. Статистика ієрархії протоколу допомагає аналізувати пакети, розбиваючи відображені дані, які належать чинному рівню OSI.

3. Сеанси обміну пакетами. Якщо ви використовуєте протокол TCP/IP або програму, яка працює із цим протоколом, ви маєте побачити чотири активних вкладок для обміну пакетами за допомогою Ethernet, IP, TCP та UDP. «Діалог» між комп'ютерами відображає трафік між двома активними хостами. Номер, зазначений на вкладці після назви протоколу, означає кількість «діалогів» між хостами. Номер, зазначений на вкладці після назви протоколу, означає кількість «діалогів» між хостами, наприклад, «Ethernet:6».

4. Точки призначення. Точки призначення забезпечують статистику даними про відправку та прийом пакетів. Номер, зазначений на вкладці після назви протоколу, вказує на кількість точок призначення. Наприклад, «Ethernet:6».

5. Графіки I/O. Основний графік може бути отриманий за допомогою команди «IO graphs» (Графіки I/O). Ще декілька графіків можуть бути додані у тому ж вікні на основі фільтрів відображення.

6. Час чекання відповіді від сервісу. 13 протоколів доступні для глибокого аналізу.

7. RTP. RTP (Real-time Transport Protocol, протокол передачі у реальному часі, RFC 3550) – це протокол для передачі звука та відео через IP-мережу. Він працює у початку протоколу дейтаграм користувача (User Datagram Protocol, UDP). Він часто використовується у сукупності з протоколами SIP або H.233, забезпечуючи виконання сигнальних завдань.

8. SIP. SIP (Session Initiation Protocol, протокол встановлення сесії, RFC 3261) – це сигнальний протокол, який оголошує відео– або VoIP-сесії. Він працює разом із протоколом RTP, який використовується для передачі мультимедійних даних.

9. Виклики VoIP.

VoIP (Voice over IP, голосовий зв'язок за допомогою Інтернету) взагалі використовує два типи протоколів:

– сигнальні протоколи, такі, як SIP або H.323

– переносні протоколи, наприклад, RTP

10 Призначення. Відображення усіх IP-адрес призначення мережевих пакетів.

11. Графік потоків. Графіки потоків забезпечує послідовний аналіз TCP-з'єднань. Перші три строки містять оголошення TCP– з'єднання з послідовностями «SYN», «SYN ACK» та «ACK».

12 HTTP. HTTP (Hypertext Transfer Protocol, протокол передачі гіпертексту) – це протокол типу «клієнт-сервер», який використовується для передачі HTML-файлів. HTTP-клієнт (у більшості випадків це web-браузер) відсилає HTTP-запит до web-серверу із полем «URL», який допомагає знайти потрібний файл. Web-сервер відповідає HTTP-пакетом та забезпечує клієнт необхідною web-сторінкою.

Меню «HTTP» містить три підменю:

– «Load Distribution» (Розподіл пакетів).

– «Packet Counter» (Лічильник пакетів).

– «Requests» (Запити).

14 IP-адреса. Відображення IP-адреси джерела або призначення мережевих пакетів.

15. Довжина пакету.

16. Тип порту. Відображення статистики у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT портів TCP або UDP.

Блок визначення топології мережі у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT

Блок визначення топології мережі у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT включає в себе наступні блоки:

– Блок використання відомостей із загальної системи моніторингу мережі у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT, а не опитування пристрою додатково.

– Блок складання списку пристроїв у мережі у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT, автоматично, ґрунтуючись на дані системи моніторингу.

– Блок побудови топології мережі у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT, за станом на задану дату й відстеження змін у топології протягом часу.

– Блок автоматичного визначення рівнів ієрархії пристроїв у мережі у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT, з виділенням периферійних, проміжних і центральних вузлів.

– Блок побудови топології мережі у системі кібербезпеки силової інфраструктури ЦОД побудованої на пристроях IoT, незалежно від використовуваної системи моніторингу й програмно-апаратних платформ;

– Блок комбінувати показників, на основі яких визначаються зв'язки між пристроями, і при їхньому обчисленні виконувати перевірку на значимість із використанням статистичних критеріїв.

Розглянувши усі блоки функціональної схеми перейдемо до розгляду діаграми взаємодії процесів, які відбуваються у системі.

3.4 Розробка діаграми процесів

Розглянемо розроблену діаграму процесів яка зображена на рисунку 3.3. Основна будова діаграми процесів полягає у графічному представленні складу сукупностей даних, що характеризуються як співвідношення різних частин кожної з сукупностей.

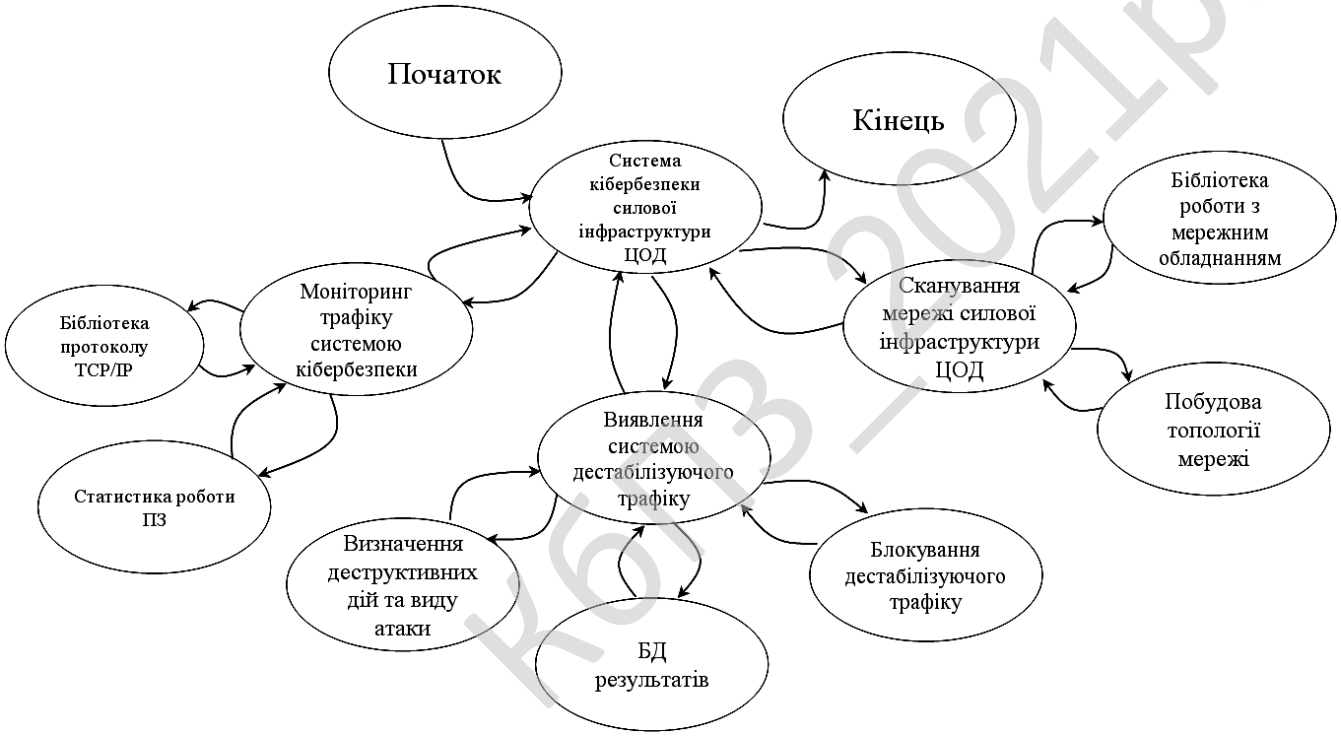

Рисунок 3.3 – Діаграма взаємодії процесів

Склад статистичної сукупності графічно може бути представлений як за допомогою абсолютних, так і відносних показників. Графічне зображення складу сукупності по абсолютними і відносними показниками сприяє проведенню більш глибокого аналізу і дозволяє проводити аналіз системи.

Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування).

Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи.

Ця діаграма в подальшому підлягає уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій системі. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

Діаграми потоків даних містять чотири типи елементів:

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

– Процеси які являють собою трансформацію даних в рамках описуваної системи.

– Сховища даних (репозиторії).

Таким чином, розглянувши опис системи, структурну, функціональну схеми системи, та діаграму взаємодії процесів перейдемо до опису блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

4 РЕАЛІЗАЦІЯ ПРОЕКТУ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Розглянемо реалізацію магістерської дипломної роботи. Були проведені розрахунки і підібрані набори тестових даних для перевірки правильності реалізації проектних рішень. Блок-схема це представлення задачі для її аналізу або розв'язування за допомогою спеціальних символів (геометричних образів), які позначають такі елементи, як операції, потік, дані тощо.

Блок вхідних та вихідних даних прийнято позначати паралелограмом, блок обчислень (обробки) даних – прямокутником, блок прийняття рішень – ромбом, еліпсом – початок та кінець алгоритму.

У інформаційних технологіях функціональна схема складається з функціональних блоків, які являють собою конструктивно відособлені частини (елементи або пристрої) автоматичних систем, які виконують певні функції. Функціональні блоки на схемі позначають прямокутниками, всередині яких надписують їх найменування відповідно до функцій, що виконуються. Зв'язки між функціональними блоками (внутрішні впливи) позначаються лініями зі стрілками, які вказують напрям впливів.

Функціональні схеми можуть виконуватися в укрупненому і розгорненому вигляді. У першому випадку на схемі зображають найважливіші блоки системи і зв'язки між ними.

У другому варіанті схема відображається більш детально, що полегшує її читання та ілюструє принцип роботи.

Основні елементи схем алгоритму це термінатор, процес, рішення, зумовлений процес (підпрограма), дані та з'єднувач. Термінатор це елемент

відображає вхід із зовнішнього середовища або вихід з неї (найчастіше застосування – початок і кінець програми). Всередині фігури записується відповідна дія.

Процес це виконання однієї або кількох операцій, обробка даних будь-якого виду (зміна значення даних, форми подання, розташування). Всередині фігури записують безпосередньо самі операції. Рішення це показує рішення або функцію перемикального типу з одним входом і двома або більше альтернативними виходами, з яких тільки один може бути обраний після обчислення умов, визначених всередині цього елементу.

Вхід в елемент позначається лінією, що входить зазвичай у верхню вершину елементу. Якщо виходів два чи три то зазвичай кожен вихід позначається лінією, що виходить з решти вершин (бічних і нижній). Якщо виходів більше трьох, то їх слід показувати однією лінією, що виходить з вершини (частіше нижній) елемента, яка потім розгалужується.

Відповідні результати обчислень можуть записуватися поруч з лініями, що відображають ці шляхи. Зумовлений процес (підпрограма) це символ відображає виконання процесу, що складається з однієї або кількох операцій, що визначені в іншому місці програми (у підпрограмі, модулі). Всередині символу записується назва процесу і передані в нього дані. Дані це перетворення у форму, придатну для обробки (введення) або відображення результатів обробки (виведення). Цей символ не визначає носія даних (для вказівки типу носія даних використовуються специфічні символи). З'єднувач це символ відображає вихід в частину схеми і вхід з іншої частини цієї схеми.

Використовується для обриву лінії та продовження її в іншому місці (приклад: поділ блок-схеми, що не поміщається на листі). Відповідні сполучні символи повинні мати одне (при тому унікальне) позначення.

На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми. З яких видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів

системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підсистеми та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ.

При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

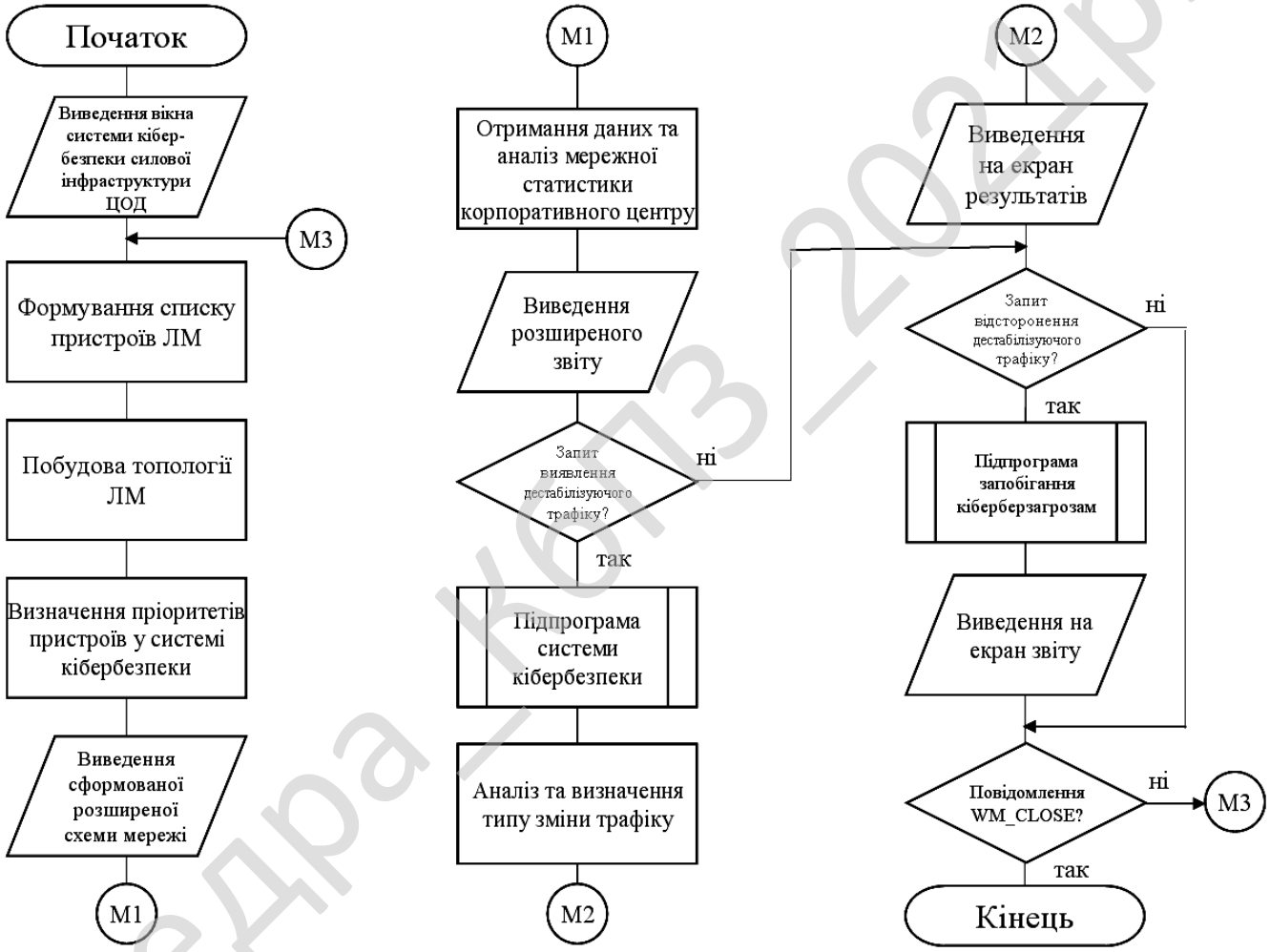

Рисунок 4.1 – Блок-схема основної програми

Архітектура клієнт-сервер є одним із архітектурних шаблонів програмного забезпечення та є домінуючою концепцією у створенні розподілених мережних програм і передбачає взаємодію та обмін даними між ними. Вона передбачає такі основні компоненти:

– набір серверів, які надають інформацію або інші послуги програмам, які звертаються до них;

– набір клієнтів, які використовують сервіси, що надаються серверами;

– мережа, яка забезпечує взаємодію між клієнтами та серверами.

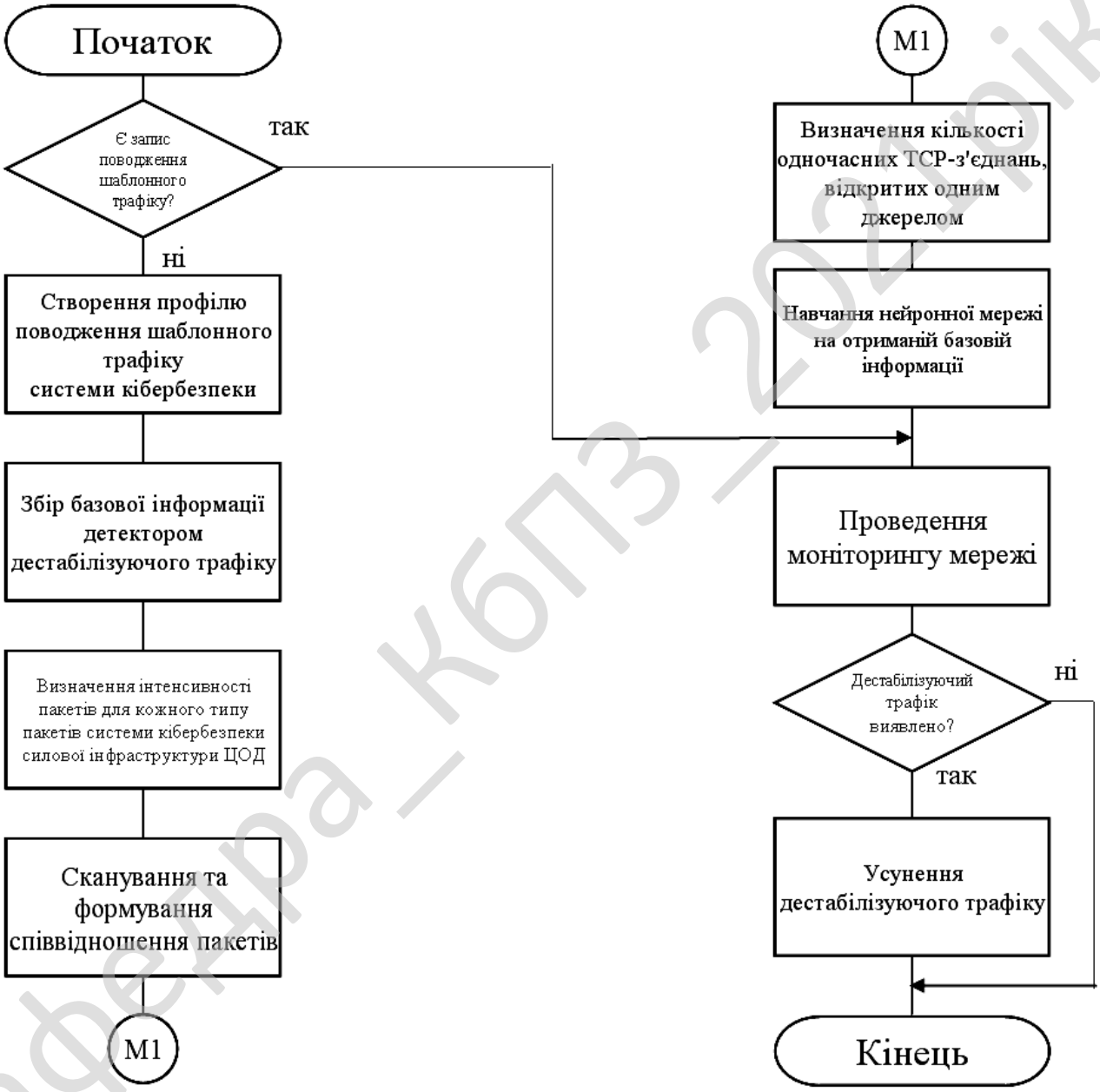

Рисунок 4.2 – Блок-схема роботи підпрограми

Сервери є незалежними один від одного. Клієнти також функціонують паралельно і незалежно один від одного. Немає жорсткої прив'язки клієнтів до

серверів. Більш ніж типовою є ситуація, коли один сервер одночасно обробляє запити від різних клієнтів; з іншого боку, клієнт мо же звертатися то до одного сервера, то до іншого. Клієнти мають знати про доступні сервери, але можуть не мати жодного уявлення про існування інших клієнтів.

Дуже важливо ясно уявляти, хто або що розглядається як «клієнт». Можна говорити про клієнтський комп'ютер, з якого відбувається звернення до інших комп'ютерів. Можна говорити про клієнтське та серверне програмне забезпечення. Нарешті, можна говорити про людей, які бажають за допомогою відповідного програмного та апаратного забезпечення отримати доступ до тієї чи іншої інформації.

Загальноприйнятим є положення, що клієнти та сервери – це перш за все програмні модулі. Найчастіше вони знаходяться на різних комп'ютерах, але бувають ситуації, коли обидві програми – і клієнтська, і серверна, фізично розміщуються на одній машині; в такій ситуації сервер часто називається локальним.

Модель клієнт-серверної взаємодії визначається перш за все розподілом обов'язків між клієнтом та сервером. Логічно можна відокремити три рівні операцій:

– рівень представлення даних, який по суті являє собою інтерфейс користувача і відповідає за представлення даних користувачеві і введення від нього керуючих команд;

– прикладний рівень, який реалізує основну логіку ПЗ і на якому здійснюється необхідна обробка інформації;

– рівень управління даними, який забезпечує зберігання даних та доступ до них.

Дворівнева клієнт-серверна архітектура передбачає взаємодію двох програмних модулів – клієнтського та серверного. В залежності від того, як між ними розподіляються наведені вище функції, розрізняють:

– модель тонкого клієнта, в рамках якої вся логіка ПЗ та управління даними зосереджена на сервері. Клієнтська програма забезпечує тільки функції рівня представлення;

– модель товстого клієнта, в якій сервер тільки керує даними, а обробка інформації та інтерфейс користувача зосереджені на стороні клієнта. Товстими клієнтами часто також називають пристрої з обмеженою потужністю: кишенькові комп'ютери, мобільні телефони та ін.

Типовим прикладом клієнт-серверної взаємодії є WWW. Існує величезна кількість веб-серверів, на яких розміщується та чи інша інформація. У найпростішому випадку ця інформація являє собою набір веб-сторінок, які можуть зберігатися на сервері у вигляді файлів, розмічених за допомогою мови розмітки HTML. Але ситуація, як правило, є складнішою; значна частина веб-ресурсів на сучасному етапі є динамічними, тобто вони не існують в заздалегідь підготовленому вигляді, а створюються безпосередньо в процесі обробки запиту від користувача.

Для того, щоб людина, яка працює в Інтернеті, могла переглянути ту чи іншу сторінку, на її комп'ютері повинно бути встановлено відповідне програмне забезпечення. Програми для перегляду веб-сторінок називаються браузерами.

Але, крім браузерів, до серверів можуть звертатися і інші клієнти, а саме – автономні програми. Вони можуть передбачати взаємодію з людиною, а можуть працювати в цілком автоматичному режимі. Типовим класом таких програм є роботи, призначені для автоматичного перегляду веб-ресурсів. Зокрема, роботи є важливим елементом пошукових систем і використовуються ними для перегляду сторінок і збору інформації про них.

Для запиту до веб-сервера клієнтська програма повинна задати місцезнаходження комп'ютера, на якому розміщується серверна програма, назву потрібного документа і, можливо, інші дані, які специфікують запит. Мережа забезпечує знаходження сервера і передачу йому клієнтського запиту. Серверні програми обробляють цей запит, відповідь пересилається по мережі клієнтові.

Трирівнева клієнт-серверна архітектура, яка почала розвиватися з середини 90-х років, передбачає відділення прикладного рівня від управління даними. Відокремлюється окремий програмний рівень, на якому зосереджується прикладна логіка ПЗ. Програми проміжного рівня можуть функціонувати під управлінням спеціальних серверів ПЗ, але запуск таких програм може здійснюватися і під управлінням звичайного веб-сервера. Нарешті, управління даними здійснюється сервером даних.

Для роботи з системою користувач використовує стандартне програмне забезпечення – звичайний браузер. Це позбавляє його необхідності завантажувати та інсталювати спеціальні програми (хоча інколи така необхідність все-таки виникає).

Але користувачеві слід надати в розпорядженні інтерфейс, який дозволяв би йому взаємодіяти з системою і формувати запити до неї. Форми, що визначають цей інтерфейс, розміщуються на веб-сторінках та завантажуються разом з ними.

Веб-оглядач формує запит та пересилає його до сервера, який здійснює обробку. При необхідності сервер викликає серверні програмні модулі, які забезпечують обробку запиту і в разі потреби звертаються до сервера даних. Сервер даних здійснює операції з даними, що зберігаються в системі та складають її інформаційну основу. Зокрема, він може здійснити вибірку з інформаційної бази відповідно до запиту та передати її модулю проміжного рівня для подальшої обробки. Дані, з якими працює сервер даних, найчастіше організовані як реляційна база даних.

Найчастіше веб-сервер і серверні модулі проміжного рівня розміщуються на одному комп'ютері, хоч і являють собою окремі і логічно незалежні програмні модулі.

На сучасному етапі для програмування модулів проміжного рівня використовується мова серверних сценаріїв PHP, а для управління даними – СУБД MySQL. Таким чином, зв'язку PHP-MySQL слід розглядати як стандартний

інструмент для створення порівняно простих інтерактивних веб-сайтів та систем електронної комерції; близько 90% комерційних систем сьогодні створюється саме на цій основі. Водночас як засоби управління даними, так і middleware-засоби можуть бути найрізноманітнішими. Так, для створення серверних програм, крім PHP, широко застосовуються Java, Perl, Python, Delphi.

Взагалі, технології створення розподілених, зокрема веб-програм, стрімко розвиваються. Слід згадати про технології EJB (Enterprise Java Beans), CORBA, а також про .NET – порівняно нову ініціативу компанії Microsoft. Для зберігання даних та їх передачі часто використовується так звана розширювана мова розмітки XML (Extensible Markup Language).

Розглянемо визначення API. Це прикладни́й програ́мний інтерфе́йс (Application Programming Interface, API) – набір визначень підпрограм, протоколів взаємодії та засобів для створення програмного забезпечення.

Спрощено – це набір чітко визначених методів для взаємодії різних компонентів. API надає розробнику засоби для швидкої розробки програмного забезпечення. API може бути для веб-базованих систем, операційних систем, баз даних, апаратного забезпечення, програмних бібліотек.

Одним з найпоширеніших призначень API є надання набору широко використовуваних функцій, наприклад для малювання вікна чи іконок на екрані. API є абстрактним поняттям – програмне забезпечення, що пропонує деякий API, часто називають реалізацією (implementation) даного API. У багатьох випадках API є частиною набору розробки програмного забезпечення, водночас, набір розробки може включати як API, так і інші інструменти/апаратне забезпечення, отже ці два терміни не є взаємозамінювані.

Високорівневі API часто програють у гнучкості. Виконання деяких функцій нижчого рівня стає набагато складнішим, або навіть неможливим.

В об'єктно-орієнтованих мовах, прикладний програмний інтерфейс зазвичай включає в себе опис набору визначень класу, з набором форм поведінки, пов'язаних з цими класами. Це абстрактне поняття пов'язане з реальними

функціями, які надані або надаватимуться, класами, які реалізуються в методах класу.

Прикладний програмний інтерфейс в даному випадку можна розглядати як сукупність всіх методів, які публічно доступні в класах (зазвичай званий інтерфейс класу). Це означає, що прикладний програмний інтерфейс вказує методи, за допомогою яких взаємодіє з об'єктами, отриманими з визначень класів і обробляє їх.

У більш загальному плані можна визначити Прикладний Програмний Інтерфейс як сукупність усіх видів об'єктів, які можна вивести з визначення класу, і пов'язаних з ними можливих варіантів поведінки.

Наприклад: клас, що представляє Stack, може просто виставити публічно два методи Push() (для додавання нового елемента в стек) і Pop() (для вилучення останнього пункту, ідеально розташований на вершині стека).

У цьому випадку Прикладний Програмний Інтерфейс може бути інтерпретованим як два методи pop() і push(), або, більш широко, використовується варіант, коли можна використовувати елемент типу Stack, який реалізує поведінку стека, надаючи йому можливість для додавання / видалення елементів з вершини. Друга інтерпретація видається більш доречною в дусі об'єктно-орієнтованого підходу.

Якість документації, пов'язаної з Прикладним Програмним Інтерфейсом, є часто ключовим фактором, що визначає його успішність з точки зору простоти використання.

ППІ, як правило, пов'язаний із бібліотеками програмного забезпечення: ППІ описує і вказує очікувану поведінку в той час, як бібліотека є фактичною реалізацією даного набору правил. Один ППІ може мати декілька реалізацій (або жодної, будучи абстрактним) у вигляді різних бібліотек, які мають такий же інтерфейс.

Прикладний програмний інтерфейс також може бути пов'язаним з платформами програмування: платформа може бути заснована на кількох

бібліотеках реалізує декілька інтерфейсів ППІ, але на відміну від звичайного використання ППІ, доступ до поведінки вбудований в платформу опосередкований шляхом розширення його змісту новими класами і вставлений в саму платформу. Крім того, загальний потік управління програми може бути під контролем абонента.

Прикладний програмний інтерфейс може бути також реалізацією протоколу.

Коли ППІ реалізує протокол, він може бути заснованим на проксі-методах віддалених викликів, що засновані на протоколі зв'язку. Роль ППІ може заключатися саме в тому, щоб приховати деталі транспортного протоколу. Наприклад: RMI є ППІ, який реалізує протокол або JRMP IIOP як RMI-IIOP.

Протоколи, як правило, розподіляються між різними технологіями і зазвичай дозволяють різним технологіям обмінюватися інформацією, діючи як абстракція між двома світами. ППІ, як правило, є специфічним для конкретної технології: звідси, інтерфейси даної мови не можуть бути використані на інших мовах, якщо виклики функції не будуть перетворені з конкретної адаптації бібліотеки.

Деякі мови, серед яких такі, що працюють на віртуальних машинах (наприклад: мови, сумісні з NET CLI середовища CLR і JVM сумісних мов у віртуальній машині Java) можуть ділитися програмними інтерфейсами.

У цьому випадку віртуальна машина дозволяє мові взаємодії завдяки спільному знаменнику віртуальної машини, що абстрагується від конкретної мови, використовувати проміжний байт-код і його мову.

При використанні прикладного програмного інтерфейсу в контексті веб-розробки, як правило, ППІ визначається набором повідомлень запиту HTTP, також визначається структура повідомлень-відповідей, зазвичай у розширенні мови розмітки XML або в форматі об'єктного запису JavaScript (JSON). У той час як прикладний програмний інтерфейс у Web історично був практично синонімом для веб-служби, останнім часом тенденція змінилась (так званий Web 2.0) на

відхід від Simple Object Access Protocol (SOAP) на основі веб-сервісів і сервіс-орієнтованої архітектури (SOA) на більш прямі передачі репрезентативного стану (REST) стилів веб-ресурсів та ресурсів-орієнтованої архітектури (ROA).

Частина цієї тенденції пов'язана з рухом Семантичного веб-ресурсу до Опису Платформ (RDF), Концепції розвитку веб-технологій інженерних онтологій. Прикладні програмні інтерфейси у Web, що дозволяють комбінувати декількома прикладними програмними інтерфейсами в нові додатки називають гібридними.

Redmine – вільне серверне ПЗ для управління проектами та відстежування помилок. До системи входить календар-планувальник та діаграми Ганта для візуального представлення ходу робіт за проектом та строків виконання. Redmine написано на мові Ruby і є ПЗ розробленим з використанням відомого веб-фреймворку Ruby on Rails, що означає легкість в розгортанні системи та її адаптації під конкретні вимоги. Для кожного проекту можна вести свої вікі та форуми.

Функціональні можливості:

– Ведення декількох проектів.

– Гнучка система доступу з використанням ролей.

– Система відстеження помилок.

– Діаграми Ганта та календар.

– Ведення новин проекту, документів та управління файлами.

– Сповіщення про зміни за допомогою RSS-потоків та електронної пошти.

– Власна Wiki для кожного проекту.

– Форуми для кожного проекту.

– Облік часових витрат.

– Налаштування власних (custom) полів для задач, затрат часу, проектів та користувачів.

– Легка інтеграція із системами керування версіями (SVN, CVS, Git, Mercurial, Bazaar и Darcs).

– Створення записів про помилки на основі отриманих листів

– Підтримка LDAP автентифікації.

– Можливість самореєстрації нових користувачів.

– Багатомовний інтерфейс (у тому числі українська мова).

– Підтримка СКБД: MySQL, PostgreSQL, SQLite.

Діаграма Ганта (*Gantt chart*, також стрічкова діаграма, графік Ганта) – це популярний тип діаграм, який використовується для ілюстрації плану, графіка робіт за будь-яким проектом. Є одним з методів планування та управління проектами.

Діаграма Ганта являє собою відрізки (графічні плашки), розміщені на горизонтальній шкалі часу. Кожен відрізок відповідає окремому завданню або підзадачі. Завдання і підзадачі, складові плану, розміщуються по вертикалі. Початок, кінець і довжина відрізка на шкалі часу відповідають початку, кінцю і тривалості завдання. На деяких діаграмах Ганта також показується залежність між завданнями.

Діаграма може використовуватися для представлення поточного стану виконання робіт: частина прямокутника, що відповідає завданню, заштриховується, відзначаючи відсоток виконання завдання; показується вертикальна лінія, що відповідає моменту «сьогодні».

Часто діаграма Ганта використовується спільно з таблицею зі списком робіт, рядки якої відповідають окремо взятій задачі, зображеній на діаграмі, а стовпці містять додаткову інформацію про задачу.

Система відстеження помилок Багтрекер – прикладна програма для допомоги розробникам програмного забезпечення (програмістам, тестувальникам тощо) враховувати і контролювати помилки, знайдені у програмах, питання щодо функціональності, рішення та оновлення, побажання користувачів, а також стежити за процесом їх виконання.

Кожному, хто розробляв програмні продукти, добре знайоме співвідношення «20/80» – останні 20 % роботи тривають 80 % часу.

Як це не парадоксально, але нічого дивного в цій пропорції немає, адже саме на завершальній стадії починається тестування проекту, коли виявляються помилки, і що більший проект, то більше буде знайдено помилок.

Водночас досить часто виявляється, що більшість цих помилок були відомі та могли бути виправлені з меншими витратами на попередніх стадіях роботи, але не були вчасно описані, а потім загубилися серед інших важливих завдань.

Отже, система відстеження помилок у найпростішому варіанті – це процес, що включає в себе виявлення помилки, її опис, виправлення і перевірку цього виправлення, тобто процес «стеження» за багом протягом всього як його життєвого циклу, так і життєвого циклу розробки в цілому.

Сукупність інформації про дефект. Головний компонент такої системи – база даних, що містить відомості про виявлені дефекти. Ці відомості можуть включати в себе:

– номер (ідентифікатор) дефекту;

– хто повідомив про дефект;

– дата і час виявлення дефекту;

– версія продукту, в якій виявлено дефект;

– серйозність (критичність) дефекту та пріоритет рішення;

– опис кроків для відтворення дефекту (неправильної поведінки програми);

– відповідальний за усунення дефекту;

– обговорення можливих рішень та їх наслідків;

– поточний стан виправлення дефекту;

– версії продукту, в якій дефект виправлений.

Крім того, розвинені системи надають можливість прикріплювати файли, які допомагають описати проблему, наприклад, дамп пам'яті або скріншот.

Використання. Основна перевага систем відстеження помилок полягає в забезпеченні чітких централізованих оглядів, запитів на розробку (включаючи помилки і виправлення) та їх стан. У корпоративному середовищі, системи відстеження помилок можуть бути використані для генерації звітів по

продуктивності програмістів виправлення помилок. Однак, це може іноді приводити до неточних результатів, тому що різні помилки можуть мати різні ступені пріоритету та серйозності, що пов'язано з складністю їх фіксації.

Життєвий цикл дефекту. Як правило, система відстеження помилок використовує той чи інший варіант «життєвого циклу» помилки, стадія якого визначається поточним станом помилки.

Типовий життєвий цикл дефекту:

1. Новий – дефект зареєстрований тестувальником.

2. Призначений – призначений відповідальний за виправлення дефекту.

3. Дозволений – дефект переходить назад у сферу відповідальності тестувальника. Як правило, супроводжується резолюцією, наприклад:

– Виправлено (виправлення включені у версію таку-то).

– Дубль (повторює дефект, що вже знаходиться в роботі).

– Не виправлено (працює відповідно до специфікації, має занадто низький пріоритет, виправлення відкладено до наступної версії тощо).

– «В мене все працює» (запит додаткової інформації про умови, в яких дефект проявляється).

4. Далі тестувальник проводить перевірку виправлення, залежно від чого дефект або знову переходить у стан «Призначений» (якщо він описаний як виправлений, але не виправлений), або у стан «Закрито».

5. Відкрито повторно – дефект знайдено знову в іншій версії.

Система може надавати адміністраторові можливість налаштування користувачі, які можуть переглядати і редагувати помилки залежно від їх стану, переводити їх в інший стан або видаляти. У корпоративному середовищі, система відстеження помилок може використовуватися для отримання звітів, що показують продуктивність програмістів при виправленні помилок. Однак, часто такий підхід не дає достатньо точних результатів через те, що різні помилки мають різну ступінь серйозності та складності. При цьому серйозність проблеми прямо не стосується складності її усунення.

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм REDOC III, який оперує з 80-бітовим блоком. Довжина ключа може змінюватися й досягати 2560 байт (204800 біт). Алгоритм складається тільки з операцій XOR над байтами ключа й відкритого тексту, перестановки й підстановки не використовуються.

1. Створюють таблицю ключів з 256 10-байтових ключів, використовуючи секретний ключ.

2. Створюють два 10-байтових блоки масок M1 і M2. M1 являє собою результат операції XOR перших 128 10-байтових ключів, а M2 – результат операції XOR других 128 10-байтових ключів.

3. Для шифрування 10-байтового блоку:

a. Виконують операцію XOR з першим байтом блоку даних і першим байтом M1. Вибирають ключ у таблиці ключів, ро зрахованої в раунді 1. Використовують обчислене значення XOR як індекс таблиці. Виконують операцію XOR з кожним, крім першого, байтом блоку даних і відповідним байтом обраного ключа.

b. Виконують операцію XOR із другим байтом блоку даних і другим байтом M1. Вибирають ключ у таблиці ключів, розрахованої в раунді 1. Використовують обчислене значення XOR як індекс таблиці. Виконаєте операцію XOR з кожним, крім другого, байтом блоку даних і відповідним байтом обраного ключа.

c. Продовжують ці дії з усім блоком даних (з 3-10 байтами), поки не буде використаний кожний байт для вибору ключа з таблиці після виконання операції XOR з ним і відповідним значенням M1. Потім виконують операцію XOR з кожним, крім використаного для вибору ключа, байтом, і ключем.

d. Повторюють етапи a-c для M2.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено розроблене у магістерської дипломної роботі система кібербезпеки силової інфраструктури ЦОД. З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні функціональні розділи: Функціональних кнопок ПЗ; Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші; Розділу обрання групи; Розділу виведення результату роботи системи; Функції представлені у графічному вигляді.

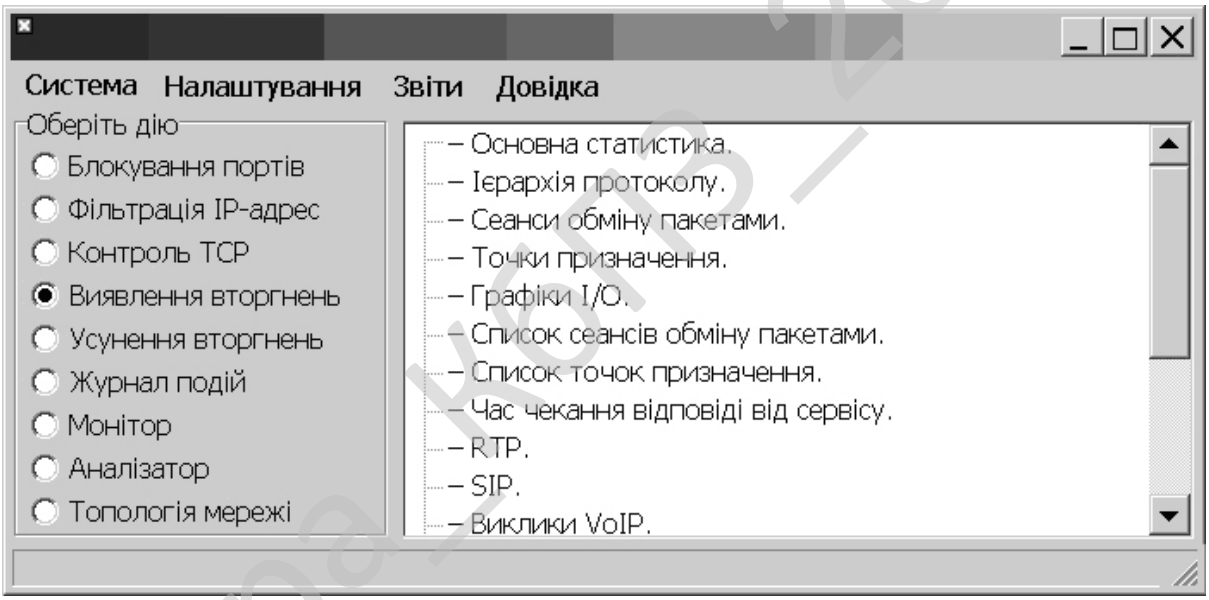

Рисунок 5.1 – Головне вікно ПЗ

Розроблена програма має дуже простий і зрозумілий інтерфейс з користувачем. Кожен, хто в достатньому обсязі володіє операційним середовищем Windows без особливих складностей освоїть і цю програму, оскільки її інтерфейс інтуїтивно зрозумілий. Якщо програма не видала ніяких помилок, і працює, то можна використовувати, інакше слід слідувати інструкціям, які пропонує програма.

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення.

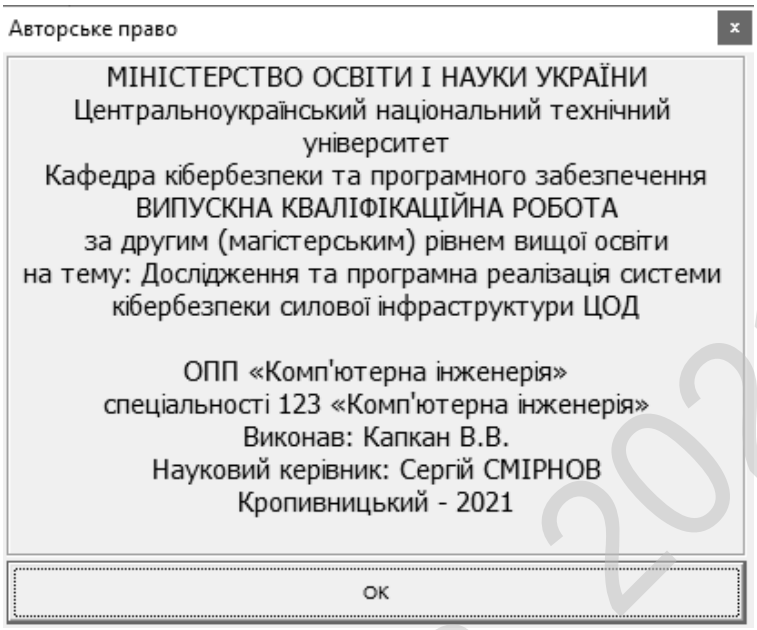

Рисунок 5.2 – Авторське право

Заражені IoT-системи, що є частиною допоміжної інфраструктури ЦОД, можуть становити небезпеку для інших мереж дати-центра. Оскільки ці пристрої пропонують точку входу в мережі центра обробки даних, ними потрібно управляти й захищати з тією же старанністю, що й сервери. Існує кілька підходів, які оператори центрів обробки даних можуть використовувати для захисту цих систем керування. Мікросегментація, наприклад, може блокувати весь трафік на пристрій, за винятком дозволеного трафіку. У деяких випадках це означає, що в кожного пристрою буде своя логічна, а не фізична мережа. Існують також спеціалізовані рішення для контролю доступу до мережі для електромереж, які активно блокують несанкціонований трафік. Коли оператор центра обробки даних здобуває нові пристрої з підтримкою IoT для комплектації ними допоміжної інфраструктури, безпека повинна бути частиною комплексної перевірки. Варто

переконатися, що паролі змінені із заводських, системи можуть бути оновлені, а настроювання пристроїв IoT – прийняті в увагу.

Розглянемо процес впровадження програмного забезпечення, це процес налаштування програмного забезпечення під певні умови використання, а також навчання користувачів роботі з програмним продуктом. Впровадження програмного забезпечення це усі дії, що роблять розроблену програмну систему готовою до використання. Даний процес є частинною життєвого циклу програмного забезпечення.

Загалом процес розгортання складається з кількох взаємопов'язаних дій із можливими переходами між ними. Ця активність може відбуватися як з боку виробника так і з боку споживача. Оскільки кожна програмна система є унікальною, то усі процеси та процедури під час розгортання важко передбачити. Тому, "розгортання" можна трактувати як загальний процес відповідно до певних вимог та характеристик. Розгортання може здійснюватись програмістом і в процесі розробки програмного забезпечення.

До діяльностей пов'язаних із розгортанням програмного забезпечення відносять:

– Випуск.

– Встановлення та активація.

– Деактивація.

– Адаптація.

– Обновлення.

– Вмонтування.

– Відстежування версій.

– Видалення.

– Вилучення з обігу.

При впровадженні програмного забезпечення потрібно урахувати наступні дії:

– Виділення критичних, з точки зору загального результату, процедур в діяльності організації. Коли набір таких процедур визначений, необхідно в першу чергу використовувати IT рішення для автоматизації операцій усередині саме цих процедур. Таким чином, розроблене IT рішення автоматично стає життєво важливим і затребуваним для організації, а також буде забезпечена публічність процесу впровадження;

– Розширення нормативної бази організації шляхом включення до неї регламентів, що описують порядок виконання процедур автоматизованих процесів. В іншому випадку є небезпека виникнення неузгодженості між автоматизованими процедурами та іншими процесами організації.

– Виконання робіт з загальної стандартизації існуючої діяльності організації, коли виділяються кращі практики виконання процедур і включаються в IT рішення за принципом найбільшої корисності для більшості учасників. Відсоток таких процедур щодо загального обсягу автоматизації може бути невеликий, але це надає процесу побудови рішення вагу в організації за рахунок збільшення його необхідності.

Під час роботи над програмою було проведено тестування програмного забезпечення, тобто технічне дослідження, призначене для виявлення інформації про якість продукту відносно контексту, в якому воно має використовуватись.

Тестування включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою їх оцінки.

Проводилась оцінка:

– відповідності поставленим вимогам;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з ОС та стороннім ПЗ.

Оскільки число можливих тестів для програмних компонент практично нескінченне, тому стратегія тестування полягала в тому, щоб провести всі можливі тести з урахуванням наявного часу та ресурсів.

Як результат ПЗ тестувалось стандартним виконанням програми з метою виявлення помилок або інших дефектів.

Проводилось тестування форматом білої скриньки засноване на аналізі керуючої структури програми. Програма вважається повністю перевіреною, якщо проведено вичерпне тестування маршрутів (шляхів) її графа управління.

У цьому випадку формуються тестові варіанти, в яких:

– Гарантується перевірка всіх незалежних маршрутів програми.

– Знаходяться гілки True, False для всіх логічних рішень.

– Виконуються всі цикли (у межах їхніх кордонів та діапазонів).

– Аналізується правильність внутрішніх структур даних.

Недоліки тестування "білої скриньки":

– Кількість незалежних маршрутів може бути дуже велика.

– Повне тестування маршрутів не гарантує відповідності програми вихідним вимогам до неї.

– У програмі можуть бути пропущені деякі маршрути.

– Не можна виявити помилки, поява яких залежить від даних.

Переваги тестування "білої скриньки" пов'язані з тим, що принцип «білої скриньки» дозволяє врахувати особливості програмних помилок:

– Кількість помилок мінімально в «центрі» і максимально на «периферії» програми.

– Попередні припущення про ймовірність потоку керування або даних у програмі часто бувають некоректними. У результаті типовим може стати маршрут, модель обчислень за яким опрацьована слабо.

– При записі алгоритму програмного забезпечення у вигляді тексту на мові програмування можливе внесення типових помилок трансляції (синтаксичних та семантичних).

– Деякі результати в програмі залежать не від вихідних даних, а від внутрішніх станів програми.

Обрано умови розповсюдження – Freeware. Це власницьке програмне забезпечення, котре можна Безоплатно використовувати протягом необмеженого терміну без обмежень у функціональності, і поширюване без сирцевих кодів.

Автори такого програмного забезпечення, як правило, хочуть «дати щось спільноті», але хочуть також контролювати його подальшу розробку. Іноді, коли програмісти вирішують припинити розробку, вони передають сирцевий код іншим програмістам, або ж спільності як вільне програмне забезпечення.

Дуже часто плутають поняття «безплатне програмне забезпечення» та «вільне програмне забезпечення», хоча вони суттєво відрізняються.

Безплатне програмне забезпечення можна безоплатно встановлювати та використовувати (іноді з певними обмеженнями, як, наприклад, «безплатне для домашнього або некомерційного вжитку»), в той час як вільне програмне забезпечення можна продавати за будь-яку суму, але при тому, у користувача, котрий його отримує, повинні бути права на вивчення, модифікацію та поширення сирцевих кодів одержаної програми.

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи кібербезпеки силової інфраструктури ЦОД.

Метою розробки є дослідження та програмна реалізація системи кібербезпеки силової інфраструктури ЦОД.

Об'єктом дослідження є процес кібербезпеки силової інфраструктури ЦОД.

Предметом дослідження є методи кібербезпеки силової інфраструктури ЦОД.

Методи дослідження базуються на методах теорії захисту інформації, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод кібербезпеки силової інфраструктури ЦОД.

– Розроблено вітчизняний продукт кібербезпеки силової інфраструктури ЦОД, який має більш широкі можливості, на відміну від існуючих аналогів.

7 ДАНІ ПРО ЕКОНОМІЧНУ ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ

7.1 Техніко-економічне обґрунтування теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

Після ознайомлення з підприємством та засобами розробки програмної продукції був розроблений план розробки програми. Був підрахований необхідний час для розробки та впровадження програми. Цей час склав 24 днів (один місяць). В магістерській роботі було проведене дослідження та виконана програмна реалізація системи кібербезпеки силової інфраструктури ЦОД.

Розроблене програмне забезпечення має достатню надійність і задовольняє усім поставленим умовам, а саме:

а) невеликий розмір;

б) невеликі системні потреби;

в) незалежність від встановлених на комп'ютері баз даних;

г) зручність у користуванні та надійність.

Таблиця 7.1 – Початкові дані

Показники	Позначення	Характеристика або величина
1	2	3
1. Кількість розроблених програм період, шт.	N	1
2. Кількість екземплярів програм, шт.	Ne	70 (ост. цифра № зал $*10^1$)
3. Запланований термін розробки, днів	Fpq	24 (1 місяць)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	Г
6. Складність алгоритму (1, 2, 3)	–	2

Продовження таблиці 7.1

1	2	3
7. Кількість макетів вхідної інформації	–	8
8. Кількість форм вихідної інформації.	–	6
9. Мова програмування (1-6)	–	2
10. Попередній досвід (1-6)	–	3
11. Гнучкість проекту ПП (1-6)	–	3
12. Детальність проекту ПП (1-6)	–	1
13. Рівень спрацьованості колективу (1-6)	–	2
14. Ступінь вимірності процесів (1-6)	–	3
15. Необхідна надійність програмного забезпечення (1-6)	–	3
16. Розмір бази даних (порівняно з розміром програми) (1-6)	–	4
17. Складність кінцевого програмного продукту (1-6)	–	5
18. Необхідний рівень забезпечення повторного використання (1-6)	–	2
19. Документованість відповідно до планованого життєвого циклу (1-6)	–	3
20. Вимоги до швидкодії ПП (1-6)	–	3
21. Обмеження на розміри основного сховища даних (1-6)	–	2
22. Різноманітність використовуваних обчислювальних платформ (1-6)	–	4
23. Професійний рівень аналітиків (1-6)	–	3
24. Професійний рівень програмістів (1-6)	–	4
25. Постійність складу команди розробників (1-6)	–	2
26. Досвід розробки додатків (1-6)	–	1
27. Досвід роботи з обчислювальною платформою (1-6)	–	2

Продовження таблиці 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	2
29. Досвід роботи з програмними інструментами розробки (1-6)	–	3
30. Розробка ПЗ для декількох серверів одночасно (1-6)	–	3
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн.	–	70000
33. Норматив додаткової зарплати, % :	$Н_д$	10
34. Норматив відрахувань у соціальні фонди, %	Нс	37
35. Норматив загальногосподарських витрат, %	Нг	15
36. Норматив витрат на освоєння нових мов програмування, %	Нп	15
37. Рівень рентабельності програмної продукції, %	Ре	30
38. Ставка податку на додану вартість, %	Ндв	20

7.2 Розрахунок трудомісткості розробки програмної продукції

Значення трудомісткості розробки програмного забезпечення для стадій ТЗ, ЕК, ТП та ВП визначаємо по типовим нормам часу приведеним в додатках МВ. Стадія РП є найбільш тривалою і трудомісткою, що робить значний вплив на інші стадії проекту.

Визначимо трудомісткість розробки ПЗ для стадії РП.

Обчислюємо номінальні трудовитрати, люд-міс.:

$$T_{ном} = A \, \text{Size}^B, \qquad (7.1)$$

де: A – коефіцієнт Боема, A = 2,45;

Size – загальний об'єм відлагодженого програмного коду, тис. рядків;

B – показник ступеня, що визначається співвідношенням:

$$B = 1{,}01 + 0{,}001\sum W_i, \quad (7.2)$$

де: *Wi* – сумарне значення п'яти показників (МВ, додаток 2), що відображають особливості розробки проекту програмного продукту (ПП) і колективу розробників.

$$B = 1{,}01+0{,}001(2{,}43+3{,}64+4{,}22+3{,}95+2{,}73) = 1{,}027.$$

$$T_{ном} = 2{,}45 \cdot 2{,}2^{1{,}027} = 5{,}5 \text{ люд-міс.}$$

Визначаємо уточнені (з урахуванням приведених в МВ додатку 3 сімнадцяти додаткових коефіцієнтів) трудовитрати, люд-міс.:

$$T_{уточн} = T_{ном}\, ПВ_j, \quad (7.3)$$

де: $ПВ_j$ – добуток сімнадцяти додаткових коефіцієнтів, приведених в МВ додатку 3.

$$T_{уточн} = 5{,}5 \cdot (1 \cdot 1{,}09 \cdot 1{,}30 \cdot 0{,}91 \cdot 1 \cdot 1 \cdot 1 \cdot 1{,}15 \cdot 1 \cdot 0{,}87 \cdot 1{,}10 \cdot$$

$$\cdot 1{,}22 \cdot 1{,}12 \cdot 1{,}10 \cdot 1 \cdot 1 \cdot 1{,}10) = 12{,}9 \text{ люд-міс.}$$

Ці коефіцієнти дозволяють диференційовано оцінювати результати роботи програмістів, беручи до уваги швидкодію програми, використання різноманітних обчислювальних платформ і інструментів розробки, взаємодію декількох серверів, вимоги до об'ємів баз даних і ін.

Визначаємо підсумкові трудовитрати по стадії робочий проект, люд-дні:

$$T_{РП} = 0{,}3 C T_{уточн}^{0{,}33+0{,}2(B-1{,}01)} S, \quad (7.4)$$

де: *C* – визначений емпірично коефіцієнт, запропонований авторами методики, (МВ, додаток 4);

S – коефіцієнт стиснення (або подовження) графіка робіт %, що дозволяє коректувати терміни розробки ПЗ згідно встановленим вимогам. Вибираємо в межах (25…350)%.

$$T_{РП} = 0{,}3 \cdot 2{,}66 \cdot 12{,}9^{0{,}33+0{,}2(1{,}027-1{,}01)} \cdot 70 = 131 \text{ люд/день.}$$

Для зручності визначення загальної трудомісткості на розробку програмного забезпечення результати розрахунків по стадіям зводимо до таблиці 7.2.

Таблиця 7.2 – Визначення трудомісткості розробки програмного забезпечення

Стадії розробки	Трудомісткість за типовими нормами та розрахунками	
	Величина, люд/дні	Підстава
Технічне завдання	9	Д5
Ескізний проект	10	Д6
Технічний проект	15	Д7
Робочий проект	131	Ф 7.1-7.4
Впровадження	15	Д13
Всього	180	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

Чисельність ставок інженерів-програмістів для розробки програмного забезпечення визначається за формулою:

$$Ч = \frac{T_{пз} N}{F_{рq} - H_{ев}}, \qquad (7.5)$$

де: $F_{рq}$ – плановий фонд робочого часу одного спеціаліста, днів;

$T_{пз}$ – трудомісткість розробки програмного забезпечення люд-дні.

$$Ч = \frac{180 \cdot 1}{24 - 3} = 8{,}6 \text{ ставки.}$$

Чисельність інженерів-електронщиків для проведення технічного обслуговування та ремонту комп'ютерних мереж визначається в залежності від наявності технічних засобів і норм витрат часу на виконання профілактичних робіт на протязі року.

Визначаємо затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за період розробки. Результати розрахунку зводимо до таблиці 7.3.

Таблиця 7.3 – Затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за розрахунковий період

Найменування обладнання	Профілактичне обслуговування			
	Кількість хв. на один. обл.	Кількість облад-нання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	385	12	4620	77
Монітор	160	12	1920	32
Клавіатура	140	12	1680	28
Маніпулятор «мишка»	30	12	360	6
Принтер матричний	185	1	185	3
Принтер лазерний	355	2	710	12
Принтер струминний	300	1	300	5
Сканер	155	2	310	5
Концентратор-маршрутизатор	155	2	310	5
Кабельні господарства ЛОМ на 1 м. п.	2,5	70	175	3
Кабельне господарство електромережі	48	50	2400	40
Копіювальний апарат	285	1	285	5
Усього за рік:			$З_ч$	221

Час на профілактику обладнання в загальному балансі робочого часу інженерів-електронщиків неповинен складати більше 10%.

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

$$Ф_{др}^{с} = \frac{З_ч \cdot n_{mic}}{1,2}, \quad (7.6)$$

$$Ф_{др}^{с} = \frac{221 \cdot 1}{1,2} = 184,1 \text{ год.}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

$$Ч_{ел} = \frac{Ф^{с}_{др}}{F_{др} \cdot T_{зм}}, \quad (7.7)$$

$$Ч_{ел} = 184{,}1/(24 \cdot 8) = 1 \text{ ставки.}$$

Для забезпечення нормального технічного обслуговування засобів ТО та мереж, необхідно прийняти найбільше ціле значення розрахункової чисельності інженерів-електронщиків.

Чисельність інженерів-системотехніків, адміністраторів мережі, дизайнерів WEB вузлів, системних програмістів (аналітиків), бухгалтерів-економістів визначається за потребою в залежності від функціональних обов'язків. Після визначення чисельності персоналу складається штатний розклад.

Таблиця 7.4 – Розрахунок чисельності штатного персоналу сектору системного та адміністративного обслуговування засобів ОТ та комп'ютерних мереж

Посада	Вид роботи	Час	К-ть штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (ОС FreeBSD), маршрутизатора Cisco, доменного контролеру Windows Server 2019, серверу доступу ADSL (ОС Linux), налаштування ADSL, VPN PPPoE, Frame Relay, Wi-Fi	2	0,5
	Налаштування і конфігурування базової станції безпровідного зв'язку (CMTS)	0,5	
	Розробка та впровадження проектів з організації зв'язку між віддаленими об'єктами, ЛОМ	0,5	
	Забезпечення цілодобової роботи зв'язку клієнтів до мережі Інтернет	1	
Всього		4	

Продовження таблиці 7.4

Посада	Вид роботи	Час	К-ть штатних одиниць
Продакт-менеджер	Презентації нової продукції, пошук каналів збуту	2	0,5
	Підтримка постійних клієнтів	1	
	Оформлення договорів, ведення тендерів	0,5	
	Контроль взаєморозрахунків з постачальниками	0,5	
Всього		4	
Дизайнер WEB	Розробка концепції оформлення та інтерфейсу сайту, оптимізація дизайну існуючих, проектує їх структуру та навігацію	2	0,5
	Створення графічних і стилістичних елементів сайту	1	
	Оформлення банерів і промо-сторінок	0,5	
	Розміщення графіки і контенту на Інтернет сторінках	0,5	
Всього		4	
Інженер верстальник	Розробка та верстка макетів рекламної продукції та технічної документації	1	0,25
	Верстка друкованих видань	0,5	
	Додрукова підготовка макетів	0,25	
	Розміщення графіки і контенту на Інтернет сторінках	0,25	
Всього		2	

Складемо штатний розклад виконавців.

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньомісячний оклад, грн.	Всього за період розробки, грн.
Керівник (ІТ-менеджер)	1	14000	14000
Продакт-менеджер	0,5	10000	5000
Інженер-програміст	8,6	8200	70520
Інженер-електронщик	1	8000	8000
Інженер-системотехнік	0,25	8000	2000
Адміністратор мережі	0,5	8000	4000
Системний програміст	0,25	8000	2000
Дизайнер WEB	0,5	8110	4055
Інженер-верстальник	0,25	7700	1925
Бухгалтер-економіст	0,5	10000	5000
Всього за період розробки	$R_{сп}$ = 13,35	-	$Ф_{роб}$ = 116500

Розрахуємо середньоденну зарплату одного виконавця:

$$З_{сд} = \frac{Ф_{роб}}{R_{сп} F_{рд}}, \quad (7.8)$$

де: $Ф_{роб}$ – загальна сума зарплати за плановий період, грн.

$$З_{сд} = \frac{116500}{13,35 \cdot 24} = 364 \; грн.$$

7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника

Балансова вартість будівель визначається з урахуванням кількості робочих місць виконавців, питомої площі на одне робоче місце, та вартості одного квадратного метра виробничої площі:

$$Б_{уд} = R_{сп}^{1} S_{у} Ц_{пл}, \quad (7.9)$$

де: $R_{сп}^{1}$ – кількість робочих місць виконавців, шт. Приймаємо 13 робочих місць;

$S_{у}$ – питома площа на одне робоче місце, $м^2$;

$Ц_{пл}$ – вартість одного квадратного метра площі, грн.

Згідно даних ТОВ науково-дослідницького консалтингового підприємства «Пекторaль» (м. Кіровоград, вул. Глинки 16) ціна одного квадратного метра площі новобудови, вік якої не перевищує 25 років, по місту складає 250…1600 $у.о./м^2$. Враховуючи, що курс у.о. складає 25 приймаємо для розрахунку вартість одного метра квадратного рівною 6560 грн./м^2. На кожне робоче місце у середньому потрібно 8 $м^2$. З урахуванням цього:

$$Б_{уд} = 13 \cdot 8 \cdot 6560 = 682240 \text{ грн.}$$

Вартість передавальних пристроїв складає 10% від вартості будівель, і у даному випадку вона складе: 68224 грн.

Балансова вартість інвентарю розраховується за нормою 3500 грн. на одне робоче місце. Тобто:

$$І_{нв} = R_{сп}^{1} \cdot Ц_{м}, \quad (7.10)$$

де: $Ц_{м}$ – ціна меблів для одного робочого місця, грн.

$$І_{нв} = 13 \cdot 3500 = 45500 \text{ грн.}$$

Балансова вартість обчислювальної техніки визначається по оптовим цінам постачальника з врахуванням витрат на транспортування.

Специфікація на обчислювальну техніку наведена в таблиці 7.7.

Дані по оптовій ціні на обладнання та комплектуючі вибирались по прайсу фірми Brain за 28.10.21 – джерело https://brain.com.ua.

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		10947
Системний блок	VINGA ADVANCED B0166 (R3M8INT.B0166)	7347
Процесор	AMD Ryzen 3, 2100GE, 4 ядра, 4 потоки Частота процесора, 3.2 ГГц, Частота в Boos 3.6 ГГц	-
Системна плата	MB AM4, Чіпсет AMD A320, 1 x Headphone 4 x USB 3.0, 3 x Audio, 1 x Microphone, 4 USB 2.0, 2 x PS/2, 1 x HDMI, 1 x VGA, 1 RJ45, Realtek ALC887, 10/100/1000 Мбіт/с	-
Відеокарта	Вбудована AMD Radeon Vega 8	-
Жорсткий диск	SSD 120 GB	-
Оперативна пам’ять	8 ГБ DIMM, DDR4-2666 MHz, PC4-21300	-
DVD-привод	Не комплектується	-
Корпус	ATX Vinga CS108B, PSU 350W(FSP Branc ATX-400PNR, 12cm), black, (front bezel black+light silver; body material – 0.6mm) 80mm fan (rear), 2xUSB2.0/AUDIO/MIC, A Duct, Tool-less chassis design,Thermall Advantaged Chassis	-

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Кардрідер внутрішній	USB 2.0 Card reader STORM CR-35U1A4-E int. 3.5", 1*USB2.0+AUDIO+1394, multi: Al Type Cards, black	220
інше	Клавіатура, мишка	Подару-нок
Монітор	PHILIPS 223V5LSB2/10, 21.5", TN, 1920 1080, 16:9, WLED, матове покриття, 5мс	3600
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струминний	Epson Stylus Photo P50 (C11CA45341) + USB cable	5500
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965

Для визначення необхідної кількості капітальних вкладень складемо таблицю 7.8.

Таблиця 7.7 – Балансова вартість обчислювальної техніки

Найменування обчислювальної техніки	Кількість, шт.	Ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Персональні комп’ютери	15	10947	16420,5	180625,5
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	199177

Витрати на транспорт, монтаж та випробування прийняті в межах до 10% від оптової ціни.

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відрахування, грн.
1	2	3	4
Група 3			
1. Будівлі	682240	-	-
2. Передавальні пристрої	68224	-	-
Всього по групі	750464	5	37523,2
Група 4			
3. Обчислювальна техніка	199177	-	-
Всього по групі	199177	50	99588,5
4. Нематеріальні активи	70000	50	35000
Група 5, 6			
5. Вимірювальні пристрої	9031	25	2257,75
6. Транспортні засоби	143000	20	28600
7. Господарський інвентар	45500	25	11375
Всього по групі	197531	-	42232,75
Разом	$К_р$ = 1217172		$А_р$ = 214344,45

Примітка: вартість автомобіля Sens (Standard+) взята по даним з автосалону «Кіровоград-Авто», джерело http://kirovograd-avto.ukravto.ua /catalog/tm-9/model-80/description, складає 143000 грн.

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців:

$$З_о = \frac{З_{сд} \cdot Т_{пз}}{N_е}, \quad (7.11)$$

де: $N_е$ – кількість екземплярів програм, шт.

$$З_о = 364 \cdot 180/70 = 936 \text{ грн.}$$

Визначимо додаткову зарплату (оплата відпусток, виконання державних та суспільних обов'язків) на рівні 10%:

$$З_д = З_о \cdot Н_q \cdot 0{,}01, \quad (7.12)$$

де: $Н_q$ – норматив додаткової зарплати, %.

$$З_д = 936 \cdot 10 \cdot 0{,}01 = 93{,}6 \text{ грн.}$$

Відрахування на соціальні потреби за нормативом $Н_с = 37\%$ від суми основної та додаткової зарплати:

$$С_{оц} = 0{,}01 \cdot Н_с (З_о + З_д), \quad (7.13)$$

де: $Н_с$ – відрахування на соціальні потреби, %.

$$С_{оц} = 0{,}01 \cdot 37(936+93{,}6) = 381 \text{ грн.}$$

Визначимо загальногосподарські витрати (електроенергію, ремонт і утримання приміщень і т.д) за нормативом $Н_г = 15\%$ від основної зарплати:

$$Г_{осп} = З_о \cdot Н_г \cdot 0{,}01, \quad (7.14)$$

де: $Н_г$ – загальногосподарські витрати, %.

$$Г_{осп} = 936 \cdot 15 \cdot 0{,}01 = 140 \text{ грн.}$$

Визначимо витрати на матеріали для розробки програмної продукції за нормами споживання та діючими цінами за одиницю виміру:

$$З_М = (З_{М1} + З_{М2} + З_{М3})/N_е, \quad (7.15)$$

де: $З_{М1}$ – вартість паперу, грн.; $З_{М2}$ – вартість запам'ятовуючих пристроїв, грн.; $З_{М3}$ – вартість фарби, картріджей, тонеру, грн.; $N_е$ – кількість екземплярів програм, шт.

Згідно виданих викладачем норм приймаємо 0,5 пачки паперу на місяць розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п = 80$ грн., визначаємо вартість паперу за період розробки $N_м = 1$ міс:

$$З_{М1} = Ц_п \cdot N_м. \quad (7.16)$$

$$З_{М1} = 80 \cdot 1 \cdot 0{,}5 = 40 \text{ грн.}$$

Згідно виданих викладачем норм до вартості запам'ятовуючих пристроїв входить вартість CD дисків в кількості, що дорівнює 30 екземплярів програм та одного DVD диска для збереження резервної копії програми:

$$З_{М2} = \sum Ц_д , \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR TDK 700Mb, 80Min, 52x Cake box – 3 грн./шт., DVD-R LG 4,7Gb, 16x speed Cake box – 3 грн./шт.

$$З_{М2} = 30 \cdot 3 + 3 = 93 \text{ грн.}$$

Згідно виданих викладачем норм одноразовій заправці підлягають усі друкуючі пристрої і становить:

$$З_{М3} = \sum Ц_з., \quad (7.18)$$

де: $Ц_з$ – вартість розхідних матеріалів друкуючих пристроїв: картридж для CANON LBP-3010 Black Canon 712 – 574 грн.; картридж для EPSON STYLUS PHOTO R390 – 558 грн.; картридж для CANON IR-1022A – LJ Q2612A Cart. HP LJ 1010/1012/1015/3015/3020/3030 (2500 стр.) – 570 грн.

$$З_{М3} = 574 + 558 + 570 = 1702 \text{ грн.}$$

$$З_М = (40 + 93 + 1702)/70 = 26 \text{ грн.}$$

Визначимо витрати на освоєння нових мов програмування або операційних систем за нормативом ($Н_п = 15\%$) від основної зарплати виконавців:

$$О_п = З_о \cdot Н_п \cdot 0{,}01, \quad (7.19)$$

де: $Н_п$ – норматив витрат на освоєння нових мов програмування, %.

$$О_п = 936 \cdot 15 \cdot 0{,}01 = 140 \text{ грн.}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е = 70$ прим.):

$$A_м = \frac{A_р \cdot N_{міс}}{N_е \cdot 12}, \quad (7.20)$$

де: $A_р$ – загальна річна сума амортизаційних відрахувань, грн.

$$A_м = 214344 \cdot 1/(70 \cdot 12) = 255 \text{ грн.}$$

Повна собівартість ПЗ визначається як сума витрат за попередніми статтями калькуляції:

$$C_п = З_о + З_д + C_{оц} + Г_{осп} + З_м + О_п + A_м. \quad (7.21)$$

$$C_п = 936+93,6+381+140+26+140+255 = 1971,6 \text{ грн.}$$

Визначимо плановий прибуток за рівнем рентабельності ($P_п$) програмної продукції, яка залежить від складності програми та ступеня новизни задачі.

Для даного програмного забезпечення рівень рентабельності складає 30%.

$$П_р = 0,01 \cdot P_п \cdot C_п, \quad (7.22)$$

де: $P_п$ – рівень рентабельності, %.

$$П_р = 0,01 \cdot 30 \cdot 1971,6 = 591 \text{ грн.}$$

Величини ціна підприємства, податок на додану вартість, відпускна ціна програмної продукції визначаються за формулами, приведеними в таблиці 7.9

Таблиця 7.9 – Нормативна калькуляція собівартості розробки програмного забезпечення задачі

Найменування статей витрат	Позначення	Величина, грн.
1	2	3
1. Основна зарплата виконавців	$З_о$	936
2. Додаткова зарплата виконавців	$З_д$	93,6
3. Відрахування на соціальні потреби	$C_{оц}$	381
4. Загальногосподарські витрати	$Г_{осп}$	140
5. Витрати на матеріали	$З_М$	26
6. Освоєння нових операційних систем, мов програмування	$О_п$	140

Продовження таблиці 7.9

1	2	3
7. Амортизація основних фондів	$А_м$	255
8. Повна собівартість програмного забезпечення	$С_п$	1971,6
9. Плановий прибуток	$П_р$	591
10.Ціна підприємства $Ц_п = С_п + П_р$	$Ц_п$	2562,6
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	512,5
12.Відпускна ціна програмної продукції $Ц = Ц_п + ПДВ$	$Ц$	3075,1

7.6 Визначення об’єму капітальних вкладень у споживача програмної продукції

Об’єм капітальних вкладень у споживача програмної продукції визначаємо на основі балансової вартості основних фондів, яка враховує ціну, транспортно-заготівельні витрати, вартість будівель, монтажних та пусконалагоджувальних робіт, а також витрати на випробування у виробничих умовах. Результати розрахунків зводимо у таблицю 7.10.

Таблиця 7.10 – Розрахунок об’єму капітальних вкладень у споживача програмної продукції

Найменування капітальних вкладень	Сума за варіантами, грн.	
	Базовий	Новий
Вартість програмної продукції	–	3075
Всього капітальних витрат	–	3075

7.7 Визначення експлуатаційних витрат

Експлуатаційні витрати у споживача програмної продукції визначаємо при умові роботи підсистеми на протязі року. Результати зводимо до таблиці 7.11.

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на обслуговування	$З_р$	28211	17632
2. Витрати на електроенергію	$З_{ел}$	421	263
3. Витрати на амортизацію	$З_{ам}$	0	1538
Всього витрат за рік	$І$	28632	19433

Витрати на обслуговування системи:

$$З_р = Т_р \cdot З_г \cdot (1 + 0{,}01 \cdot Н_q) \cdot (1 + 0{,}01 \cdot Н_c), \qquad (7.23)$$

де: $Т_р$ – кількість годин обслуговування за рік, *год.*;

$З_г$ – заробітна плата обслуговуючого персоналу, *грн / год*.

Після купівлі нового програмного забезпечення кількість профілактичних годин робіт зменшилася з 240 годин на рік до 150 годин на рік, тому витрати на технічне обслуговування зменшилися з:

$$З_{р\,баз} = 240 \cdot 26 \cdot 1{,}1 \cdot 1{,}37 \cdot 3 = 28211\ грн,$$

до:

$$З_{р\,нов} = 150 \cdot 26 \cdot 1{,}1 \cdot 1{,}37 \cdot 3 = 17632\ грн.$$

Витрати на електроенергію визначаються з урахуванням споживаємої потужності ($П_{ел}$) в кіловатах, часу експлуатації технічних засобів ($Т_р$) в годинах та ціни однієї кіловат-години ($Ц_{ел}$):

$$З_{ел} = П_{ел} \cdot Т_р \cdot Ц_{ел}. \qquad (7.24)$$

$$З_{ел\ баз} = 7 \cdot 0{,}15 \cdot 240 \cdot 1{,}67 = 421 \text{ грн.}$$

$$З_{ел\ нов} = 7 \cdot 0{,}15 \cdot 150 \cdot 1{,}67 = 263 \text{ грн.}$$

Витрати по амортизації визначаються на основі норм амортизаційних відрахувань, вартості програмної продукції і основних фондів. Для розрахунку складаємо таблицю 7.12.

Таблиця 7.12 – Розрахунок амортизаційних відрахувань

Групи основних фондів	Норма амортизації %	Балансова вартість, грн., за варіантами		Сума відрахувань, грн за варіантами	
		Базовий	Новий	Базовий	Новий
Програмна продукція	50	–	3075	–	1537,5
Всього відрахувань	-	–	3075	–	1537,5

7.8 Визначення економічної ефективності програмної продукції

Економічна ефективність програмного забезпечення визначається для виготовлювача і споживача за такими показниками.

Величина економічного ефекту при виготовленні програмної продукції, розраховуємо за формулою:

$$E_в = (Ц_п - C_п) \cdot N_е - E_н \cdot K_р, \quad (7.25)$$

де: $K_р$ – балансова вартість основних фондів розробника, грн.

$$Ев = (2562,6 - 1971,6) \cdot 70 - 0,15 \cdot 1217172 \cdot 1/12 = 26155,35 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

$$T_в = \frac{K_р}{(Ц_п - C_п) \cdot N_е}, \quad (7.26)$$

$$T_в = \frac{1217172}{(2562,6 - 1971,6) \cdot 70 \cdot 12 / 1} = 2,5 \text{ року}.$$

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	70
2. Повна собівартість розробленої програми	Грн.	1971,6
3. Ціна розробленої програми	Грн.	2562,6
4. Плановий прибуток від реалізації розробленої програми	Грн.	591
5. Рентабельність програмної продукції	%	30
6. Об'єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1217172
7. Загальний прибуток від реалізації програмної продукції	Грн.	41370
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	26155
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років	2,5
10.Об'єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	3075
11.Величина економічного ефекту у користувача програмної продукції	Грн.	7662
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	0,3

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

$$E_{сп} = (I_б - I_н) - E_н(K_н - K_б), \qquad (7.27)$$

де: $I_б$, $I_н$ – величина експлуатаційних витрат за базовим и новим варіантом відповідно;

$К_б$, $К_н$ – об'єм капітальних вкладень за варіантами, що порівнюються.

$$E_{сп} = (28632\text{-}19433)\text{-}0{,}5 \cdot 3075 = 7661{,}5 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у споживача програмної продукції за рахунок зниження експлуатаційних витрат:

$$T_{сп} = \frac{К_н - К_б}{І_б - І_н}, \tag{7.28}$$

$$T_{сп} = \frac{3075}{28632 - 19433} = 0{,}3 \ року.$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

7.9 Висновки

Розроблена програма економічно вигідна. За рахунок впровадження програмного забезпечення досягається скорочення часу обробки інформації, підвищується культура праці, підвищення якості приймаючих управлінських рішень.

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Протягом усієї історії людство приділяє прискіпливу увагу безпеці життя. Охорона праці є складовою частиною безпеки життя [1].

Законом України “Про охорону праці” [2] регламентуються загальні положення державної політики в галузі охорони праці, а конкретизуються ці положення нормативно-правовими актами про охорону праці, зокрема Наказом Міністерства соціальної політики України 14.02.2018 № 207, який зареєстровано в Міністерстві юстиції України 25 квітня 2018 р. за №508/31960 «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями», НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин», та ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин».

Програмісти у процесі роботи мають негативний вплив на органи зору, а також мають значну розумову напругою і нервово-емоційне навантаження. Руки (суглоби польців та м'язи рук) при роботі з клавіатурою мають теж ic тотне навантаженням. До шкідливих факторів, які впливають на робитників галузі інформаційних технологій (ІТ) спеціалісти відносять високочастотні електромагнітні коливання (випромінювання) роботи апаратної частини ЕОМ та виділення шкідливих газів.

Ці шкідливі фактори можуть привести до професійних захворювань.

При розгляді шкідливих чинників роботи програмістів та інших спеціалістів ІТ будемо керуватись наступними нормативно-правовими актами: «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98, та

«Правила охорони праці під час експлуатації електронно-обчислювальних машин» НПАОП 0.00-1.28-10,

Умови праці програміста вулючають наступні фактори:

– параметри повітряного середовища в приміщенні;

– вентиляція приміщення;

– освітлення приміщення;

– параметри повітряного середовища в приміщенні, тощо.

Щоб запропонувати заходи щодо зменшення впливу комп'ютера на організм програміста визначемо фактори, які можуть викликати професійне захворювання і впливають на працездатність програміста,

8.2 Шкідливі і небезпечні фактори при роботі з комп'ютером

Програміст працює з електронно-обчислювальною машиною (ЕОМ) та іншим обладнанням, яке є джерелом небезпеки ураження електричним струмом. Так як робота програміста характеризується істотним зоровим навантаженням, то вимагає належного освітлення. Так як програміст постійно перебуває в приміщенні, тому для комфортних умов праці в цьому приміщенні необхідно створити належний мікроклімат.

При роботі з використанням ЕОМ відзначають наступні небезпечні та шкідливі фактори:

– ризик виникнення надзвичайних ситуацій природного або штучного характеру на об'єкті або території.

– ризик виникнення пожежі;

– негативний вплив на органи зору людини;

– ризики ураження електричним струмом;

– недостатня, або надмірна освітленість робочого місця;

– монотонність праці;

– електромагнітні (у т.ч. високочастотні) випромінювання (коливання);

– несприятливі мікрокліматичні умови;

– нервово-емоційна напруженість праці;

– інтелектуальні навантаження;

– невідповідність ергономічних показників робочого місця діючим вимогам;

– шуми;

– статичні навантаження на кістково-м’язовий апарат;

8.3 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста

Розглянемо умови праці у приміщенні, в якому працюють програмісти. Геометричні розміри приміщення наведено у таблиці 8.1.

Таблиця 8.1 – Розміри приміщення

Найменування	Значення, м
Ширина	2,4
Довжина	3
Висота	2,8

Таблиця 8.2 – Площа та обсяг приміщення, на одного працюючого

Геометрична характеристика	Одиниця виміру	Нормативне значення*	Фактичне значення
Площа, S	$м^2$	не менше 6.0	7,2
Обсяг, V	$м^3$	не менше 20.0	20,1

* Згідно ДСанПіН 3.3.2.007-98 (Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин).

У зазначеному приміщенні працє 1 людина. За даними, які наведено у табл. 8.1 та табл. 8.2, можна зробити висновок, що площа та об'єм приміщення у розрахунку на одно робоче місце програміста відповідають нормативним вимогам (Наказу Міністерства соціальної політики України № 207, від 14.02.2018 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями», ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуа льними дисплейними терміналами електронно-обчислювальних машин» та НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин»).

Температура повітря в приміщенні визначається впливом температури зовнішнього повітря і тепловою енергією, яка виділяється всередині приміщення. Джерелами виділення теплоти в даному приміщенні є електроустаткування, освітлювальні прилади, а також люди. У світлий час доби джерелом надлишкового тепла є сонячна радіація. Згідно Постанови № 42 від 01.12.1999 Головного державного санітарного лікаря України, робота, яка виконується в даному приміщенні, відноситься до категорії Іа. В цьому випадку людина витрачає енергії до 120 ккал у годину. Вологість повітря у приміщенні визначається впливом багатьох факторів, серед яких: вологість атмосферного повітря, виділення вологи людьми (при диханні та випарами з поверхні шкіри).

Мікроклімат повітряного середовища в приміщенні характеризується запиленістю та загазованістю повітря. Мікроклімат приміщення визначається діючим на організм людини поєднанням, вологості, температури, швидкості руху повітря та інтенсивності теплового випромінювання. Аналіз мікроклімату складається з визначення зазначених вище факторів і порівняння результатів із встановленими нормами.

Таблиця 8.3 – Оптимальні і фактичні значення параметрів мікроклімату

Пора року	Оптимальні для Іа			Фактичні		
	Температура, °С	Воло-гість,%	Швидкість повітря, м/с	Температура, °С	Воло-гість%	Швидкість повітря, м/с
Холодна	22-24	40-60	0,1	22-23	41-59	0,1
Тепла	23-25	50-70	0,1	23,5-24	53-70	0,13

У таблиці 8.3 наведено оптимальні та фактичні значення параметрів мікроклімату як для категорії ваги робіт Іа, так і розглянутого приміщення. У приміщеннях, де встановлено ЕОМ, рекомендується застосування тільки оптимальних значень показників мікроклімату.

Проведений аналіз показує, що показники мікроклімату в приміщенні відповідають установленим нормам. Штучне опалення застосовується у холодний період року.

В літню пору застосовується кондиціонер.

Для боротьби з пилом робляться регулярні провітрювання та вологі прибирання приміщенні.

У приміщенні знаходяться наступні джерела шуму: принтер *HP LaserJet 1018*, електродвигуни вентиляторів ЕОМ.

Одним з найважливіших факторів, які впливають на ефективність трудової діяльності людини, та попереджають травматизм і професійні захворювання програмістів є освітлення на робочому місці.

Працю працівника, який постійно працює за комп’ютером, згідно ДБН В.2.5 – 28 – 2006 р. можна віднести до роботи з малою точністю (найменший розмір об'єкта розрізнення від 1 до 5 мм) V-го розряду зорової роботи, з великою контрастністю об'єкта розрізнення (символів на екрані дисплея), з темним тлом (під розряд зорової роботи В). Приміщення можна віднести до 1-ої групи

приміщень, у яких проводиться розрізнення об'єктів зорової роботи при фіксованому напрямку лінії зору того, що працює на робочу поверхню. Для такого типу приміщень і розряду зорової роботи нормоване значення коефіцієнта природної освітленості (КПО) робочої поверхні (при поєднаному, спільному освітленні), повинен становити не більше 1,5%, освітленість при штучному висвітленні повинна становити 300 лк. Крім того все поле зору повинне бути освітлено достатньо рівномірно – ця основна гігієнічна вимога. Так як яскраве світло на ділянці периферійного зору значно збільшує напруженість очей і, як наслідок, призводить до їх швидкої стомлюваності, ступінь освітлення приміщення і яскравість екрану комп'ютера повинні бути приблизно однаковими.

8.4 Розробка заходів з умов поліпшення охорони праці

Згідно аналізу умов праці в розглянутому приміщенні, ми одержали наступні результати:

– розмірі приміщення, у розрахунку на одному працюючого, відповідають нормативам;

– мікроклімат відповідає нормативному значенню;

– акустичні умови роботи не перевищують нормативних значеннь;

Таким чином можна припустити, що основною причиною можнивого зниження працездатності програміста є психофізіологічний фактор, тому основна пропозиція буде така: дотримання позитивної психологічної атмосфери в колективі та регламентованого режиму праці та відпочинку, організація робочого місця з урахуванням ергономічних вимог.

Рекомендовані заходи: регулярні періодичні наочні огляди персоналом шляхів для евакуації людей із приміщення, відповідно до плану евакуації (який повинен розташовуватись на видному місці у приміщенні), включення до колективного договору мінімально можливого вмісту аптечок з обов’язковою наявністю масок-клапанів, або іншого спорядження для штучного дихання.

Регулярна періодична перевірка параметрів заземлення та занулення (вимірювання опору розтіканню електричного струму на землю).

Так як при ураженні електричним струмом у людини може статися фібриляція шлуночків серця, в організації бажано мати дефібрилятор і підготовлений персонал для роботи з ним.

8.5 Розрахункова частина

Початкові данні для розрахунку штучного захисного заземлення:

Тип заземлення: робоче заземлення нульової точки трансформатора. Напруга – 220/380 В. Розташування заземлюючих електродів – по контуру.

Розрахунок проводиться за допустимим опором розтіканню струму заземлювача методом коефіцієнта використання заземлювачів.

Початкові дані для розрахунку захисного заземлення: тип верхнього шару грунта – чорнозем, нижнього шару грунта – глина. Умовна товщина верхнього шару грунта: $H=0,8$ *м*. Для захисного заземлення: застосовуються вертикальні електроди – пруток довжиною $L=2$ *м.* Відстань між вертикальними заземлювачами (електродами) $A=2$ *м*. Діаметр вертикального електрода (прутка) $D=60$ *мм*, Тип горизонтального заземлювача: металева полоса. Розміри перетину з'єднуючої полоси: *60 х 6 мм*. ($b=60$ *мм*.). Опір заземлювача, який нормується: $R_{ЗН} = 4$ *Ом*. Глибина закладення горизонтального контура заземлення $t= 0,6$ *м*.

Розрахунок захисного заземлення можна автоматизувати за допомогою програми, сирцевий код якої опубліковано на стр.13-16 [4].

Розрахунок.

Відстань від центра вертикального заземлювача до поверхні землі:

$$T=t+L/2=0,6+2/2=1,6 \text{ м.}$$

Еквівалентний питомий опір грунта:

$$\rho_{екв}= \psi \rho_1 \rho_2 L /[\rho_1 \psi(L-H+t)+ \rho_2 \psi(H-t)]=93,75 \text{ Ом}$$

де

$\psi = 1,2$ – табличне значення коефіцієнта сезонності для відповідної климатичной зони у багатошаровому грунті (джерело: http://ftemk.mpei.ac.ru/bgd/_private/Grunt_/t_goda.htm);

$\rho_1 = 50$ *Ом*м.* – табличне значення питомого опору верхнього шару грунта (джерело: https://el-line.ru/udelnoe_soprotivlenie_grunta.shtml);

$\rho_2 = 100$ *Ом*м.* – табличне значення питомого опору нижнього шару грунта (джерело: https://zandz.com/ru/udelnoe_soprotivlenie_grunta).

Опір розтіканню електричного струму одного електрода вертикального заземлювача:

$$R_O = [\rho_1/(2\pi*L)]*[ln(2L/D)+0,5ln(4T+L)/(4T-L)]=40,6 \text{ Ом}.$$

Опір розтіканню електричного струму заземлювача, який нормується:

$R=1$ якщо $\rho_{екв} < 100$ (*у нас* $\rho_{екв} = 93,75 < 100$),

та $R = R_{ЗН} * \rho_{екв}/100$, *якщо* $\rho_{екв} > 100$.

Опір розтіканню електричного струму горизонтального заземлювача (полоси):

$$R_П = 0,366(\rho_{екв}\ \psi / L_П \cdot \eta_П)\lg(2/ L_П * L_П/bt)=35,7 \text{ Ом}.$$

де $\eta_П = 0,3$ – табличне значення коефіцієнта використання горизонтального заземлювача, залежить від розташованя (в ряд або по контуру) та співвідношення *A/L*;довжина горизонтального заземлювача (полоси) при розташуванні заземлювачів по контуру:

$$L_П = A*n = 2*16 = 32 \text{ м.};$$

де n – ітераційна кількість вертикальних заземлювачів.

Загальний опір штучного заземлюючого пристрою розтіканню електричного струму на землю:

$$R_B = R_П\, R / (R_П - R) = 4,5 \text{ Ом}.$$

Враховуючи, що значення ρ_1 *та* ρ_2 по факту можуть бути меншими, розрахункове значення R_B цілком прийнятне.

Кількість вертикальних заземлювачів:

$$n = R_O / R_B * \eta_B = 16 \text{ шт}.$$

де $\eta_B = 0,6$ – табличне значення коефіцієнта використання вертикального заземлювача, залежить від розташованя (в ряд або по контуру) та співвідношення A/L [3].

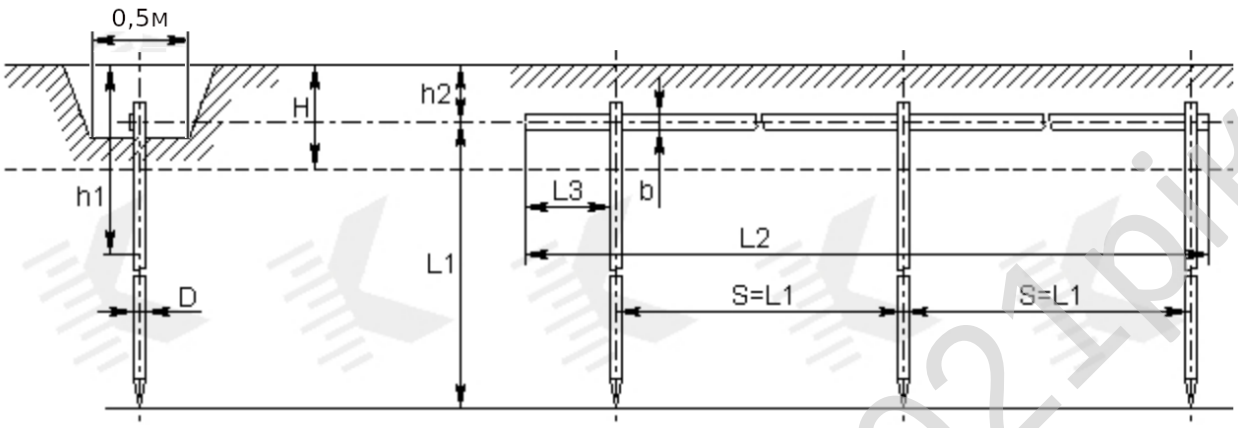

Рисунок 8.1 – Штучний заземлювач

8.6 Висновки до розділу

Дотримання всіх необхідних умов праці не лише сприяє збереженню здоров'я працівників, а також підвищує ефективність виробництва вцілому.

З цих міркувань було здійснено аналіз приміщення, призначеного для праці програмістів, проведено розгляд небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи. Виконано розрахунок штучного освітлення, як одного з ключових факторів впливу на працездатність та здоров'я програміста. Розроблено заходи з охорони праці.

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи кібербезпеки силової інфраструктури ЦОД.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів кібербезпеки силової інфраструктури ЦОД.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем кібербезпеки силової інфраструктури ЦОД.

– Досліджена система кібербезпеки силової інфраструктури ЦОД.

– На основі отриманих результатів досліджень створена програмна реалізація системи кібербезпеки силової інфраструктури ЦОД.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання кібербезпеки силової інфраструктури ЦОД.

Проведено аналіз предметної галузі в ході якого були виявлені об'єкти, взаємодія яких носить істотний характер для функціональної діяльності предметної галузі, і їхні основні характеристики; побудована алгоритм і вибраний середовище розробки.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створені програмного забезпечення було використано об’єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня Embarcadero RAD Studio Delphi 10.3.2 Rio Architect. Дана мова програмування дозволяє найбільш ефективно обробляти дані. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows XP/Vista/7/8/10.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм REDOC III.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

Розроблена програма має реальний економічний ефект від її впровадження у виробництво у сумі 7662 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 0,3 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Капкан В.В. Дослідження та програмна реалізація системи кібербезпеки силової інфраструктури ЦОД // Збірник праць молодих науковців ЦНТУ. – Вип. 12. – Кропивницький: ЦНТУ, 2022.

2. Комплексный подход к построению интеллектуальной системы предотвращения деструктивным действиям дестабилизирующего трафіку / Васильев В. И., Свечников Л. А., Кашаев Т. Р. // Системы управления и информационные технологии, Воронеж, №2, 2007. – С. 76-82.

3. Будько М.Ю. Повышение безопасности работы компьютерних сетей на основе анализа потоков данних // Известия высших учебних заведений. Приборостроение. Том 52. Номер выпуска 5. – СПб.: СПбГУ ИТМО, 2009. – С. 31-34.

4. Будько М.Ю., Будько М.Б. Отслеживание изменений в структуре сети и решение задач повышения безопасности на основе анализа потоков данних // Научно-технический вестник Санкт-петербургского государственного университета информационних технологий, механики и оптики. Номер выпуска 59. – СПб.: СПбГУ ИТМО, 2009. – С. 78-82.

5. Будько М.Ю., Будько М.Б. Определение источника широковещательного шторма на основе данних протокола SNMP // Сборник трудов конференции молодих ученых. Выпуск 6. Информационні технологии. – СПб.: СПбГУ ИТМО, 2009. – С. 153-157.

6. Даниленко Д.О. Дисперсійний аналіз мережного трафіку для забезпечення інформаційної безпеки телекомунікаційних систем / О.О. Кузнецов, О.А. Смірнов, Д.О. Даниленко // Інформаційна та економічна безпека: сучасний стан та тенденції розвитку : монографія за заг. ред. – Х.: ХІБС УБС НБУ – 2014 – С. 82-100.

7. Даниленко Д.О. Дослідження методів виявлення вторгнень в телекомунікаційні системи та мережі / Д.О. Даниленко, О.А. Смірнов, Є.В. Мелешко // Системи озброєння і військова техніка. – Випуск 1(29) – Х.: ХУПС – 2012. – С. 92-100

8. Даниленко Д.А. Метод обнаружения вредоносного программного обеспечения. Часть 1. Корреляционный анализ сетевого трафика // А.А.Смирнов, Д.А. Даниленко, Е.В.Мелешко // Науково-технічний журнал «Інформаційно-керуючі системи на залізничному транспорті» – Випуск 4(95). – Х.: УкрДАЗТ – 2012. – С. 8-14.

9. Даниленко Д.А. Методы обнаружения вредоносного программного обеспечения в телекоммуникационных системах и сетях / Д.А. Даниленко // Збірник наукових праць "Системи обробки інформації". – Випуск 3(101) том 2. – Х.: ХУПС – 2012. – С. 152-155.

10. Даниленко Д.А. Системы обнаружения и предотвращения вторжений для защиты телекоммуникационных сетей от вредоносного программного обеспечения / Д.А. Даниленко, А.А. Смирнов, А.В. Коваленко // Системи управління, навігації та зв'язку. – Випуск 1 (21) том 2. – Київ: ДП «ЦНДІНУ». – 2012. – С. 183-186.

11. Даниленко Д.А. Системы обнаружения и предотвращения вторжений для защиты компьютерных сетей от вредоносного программного обеспечения / Д.А. Даниленко, А.А. Смирнов, И.Г. Кирилов // Збірник тез доповідей науково-практичної конференції «Застосування інформаційних технологій у підготовці та діяльності сил охорони правопорядку». м. Харків. 21-22 березня 2012 р. – Харків. АВВ МВС. – 2012. – С. 70-71.

12. ГОСТ Р ИСО/МЭК 13335-1-2006. Информационная технология. Методы и средства обеспечения безопасности. Часть 1. Концепция и модели менеджмента безопасности информационных и телекоммуникационных технологий [Електронний ресурс]. – Режим доступу до ресурсу: http://www.rfcmd.ru/sphider/docs/InfoSec/GOST-R_ISO_IEC_13335-1- 2006.htm

13. ГОСТ Р ИСО/МЭК 27033-1-2011 Информационная технология. Методы и средства обеспечения безопасности. Безопасность сетей. Часть 1. Обзор и концепции [Електронний ресурс]. – Режим доступу до ресурсу: http://protect.gost.ru/document.aspx?control=7&id=179072

14. ДСТУ В 3265 – 95. Зв'язок військовий. Терміни та визначення. – К.: УкрНДІССІ, 1995. – 23 с.

15. ДСТУ ISO 9000:2007 Системи управління якістю. Основні положення та словник термінів [Електронний ресурс]. – Режим доступу до ресурсу: http://document.ua/docs/tdoc14237.php

16. Дымарский Я.С. Управление сетями связи: Принципы, протоколы, прикладные задачи / Я.С. Дымарский, Н.П. Крутякова, Г.Г. Яновский. –М.: ИТЦ «Мобильные коммуникации», 2003. – 384 с.

17. Ершов В.А. Мультисервисные телекоммуникационные сети / В.А. Ершов, Н.А. Кузнецов – М.: Изд. МГТУ им. Н.Э. Баумана, 2003. – 432 с.

18. Защита информации в автоматизированных системах обработки информации [Електронний ресурс]. – Режим доступу до ресурсу: http://www.xserver.ru/computer/raznoe/bezopasn/2/

19. Информационное сообщение об утверждении требований к средствам антивирусной защиты от 30 июля 2012 г. № 240/24/3095 [Електронний ресурс]. – Режим доступу до ресурсу: http://fstec.ru/component/attachments/download/402

20. Касперский Е. Компьютерное зловредство / Е. Касперский. – СПб.: Питер, 2007. – 208 с.

21. Касперски К. Техника сетевых атак. [Електронний ресурс]. – Режим доступу до ресурсу: http://rghost.ru/download/43730077/

22. 8e48b6263ce45c7dc2a65a7453383dc33b22486d/Крис%20Касперски%20-%20Техника%20сетевых%20атак.pdf

23. Касперски К. Техника и философия хакерских атак / К. Касперски. – М.: Солон-Пресс, 2004 – 272 с.

24. Касперски К. Записки исследователя компьютерных вирусов / К. Касперски. – СПб.: Питер, 2006. – 316 с.

25. Касперски К. Компьютерные вирусы изнутри и снаружи. / К. Касперски. – СПб.: Питер, 2006. – 526 с.

26. Кингман Дж. Пуассоновские процессы / Дж. Кингман. – М.:МЦНМО, 2007. – 136 с.

27. Корченко А.Г. Построение систем защиты информации на нечетких множествах. Теория и практические решения / А.Г. Корченко. – К.: «МК-Пресс», 2006. – С. 207-214.

28. Конахович Г.Ф. Сети передачи пакетных данных / Г.Ф. Конахович, В.М.Чуприн. – К.: МК-Пресс, 2006. – 272 с.

29. Кучук Г.А. Управление ресурсами инфотелекоммуникаций / Г.А. Кучук, Р.П. Гахов, А.А. Пашнев. – М.: Физматлит, 2006. – 220 с.

30. Лавренков Ю.Н. Исследование и разработка комбинированных нейросетевых технологий для повышения эффективности безопасной маршрутизации информации в сетях связи: диссертация ... кандидата технических наук: 05.13.17 / Лавренков Юрий Николаевич, 2014.– 208 с.

31. Лазарев И. А. Композиционное проектирование сложных агрегативных систем / И.А. Лазарев. – М.: Радио и связь, 1986. – 312 с.

32. Лукацкий А.В. Обнаружение атак / А.В. Лукацкий. – С.Пб.: BHV, 2001. – 624 с.

33. Майника Э. Алгоритмы оптимизации на сетях и графах: пер. с англ. / Э. Майника; под ред. Е.К. Масловского. – М.: Мир, 1981. – 321 с.

34. МСЭ-Т Рекомендация G.101. Международные телефонные соединения и цепи – Общие определения //11/2003. [Електронний ресурс]. – Режим доступу до ресурсу: http://www.telecom61.ru/SharedFiles/ Download.aspx? …pageid=106

35. Назаров А.Н. Модели и методы расчета структурно-сетевых параметров АТМ сетей / Алексей Николаевич Назаров. – М.: Горячая линия – Телеком, 2002. – 256 с.

36. Одом Ш. Коммутаторы CISCO / Ш. Одом, Х. Ноттингем – М.: "Кудиц-Образ", 2003. – 528 с.

37. Олифер В.Г. Компьютерные сети. Принципы, технологии, протоколы: учебник для вузов / В.Г. Олифер, Н.А. Олифер. – 2-е изд. – СПб.: Питер, 2007. – 958 с.

38. Партыка С.А. Метод ускоренной коррекции SPT с использованием динамических алгоритмов [Електронний ресурс]. – Режим доступу до ресурсу: http://openarchive.nure.ua/bitstream/123456789/936/1/ASU_158_2012%20%2842-47%29.pdf

39. Руководство по технологиям объединенных сетей. 4-е изд. / пер.с англ. и ред. А.Н. Крикуна – М.: Изд. дом «Вильямс», 2005. – 1040 с.

40. Семенов С.Г. Математическая модель мультисервисного канала связи на основе экспоненциальной GERT-сети / С.Г. Семенов, Є.В. Мелешко, Я.В. Ілюшко // Системи озброєння і військова техніка. – Х.:ХУ ПС. – 2011. –Вип. 3(27). – С. 64-67.

41. Семенов С.Г. Математична модель системи криптографічного захисту електронних повідомлень на основі GERT-мережі / С.Г. Семенов, О.О. Сур // Системи управління, навігації та зв'язку. – К.: ЦНДІ навігації і управління. – 2012. – Том 1. Вип. 1(21). – С. 131-137

42. Семенов С.Г. Исследования вероятностно-временных характеристик мультисервисного канала связи с использованием математического аппарата GERT-сети / С.Г. Семенов, В.В. Босько, І.А. Березюк // Системи обробки інформації. – Х.: ХУ ПС, 2012. – Т. 1., Вип. 3(101). – С. 139-142.

43. Семенов С.Г. Моделирование защищенного канала связи с использованием экспоненциальной GERT-сети / С.Г. Семенов, А.А. Можаев // Информатика, математическое моделирование, экономика. – Смоленськ.: Смоленский филиал АНО ВПО ЦС РФ "Российский университет кооперации". – 2012. – Том.1. – С. 152-160.

44. Семенов С.Г. Методика математического моделирования защищенной ИТС на основе многослойной GERT-сети / С.Г. Семенов // Вісник Національного технічного університету «Харківський політехнічний інститут». – Х.:НТУ «ХПІ». – 2012. –№62 (968). – С 173-181.

45. Семенов С.Г. Защита данных в компьютеризированных управляющих системах / С.Г. Семенов, В.В. Давыдов, С.Ю. Гавриленко. – LAP Lambert Academic Publishing GmbH & Co. KG (Саарбрюккен, Германия), 2014. – 236 с.

46. Смирнов А.А. Анализ и сравнительное исследование перспективных направлений развития цифровых телекоммуникационных систем и сетей / А.А.Смирнов, В.В.Босько, Е.В.Мелешко // Системи обробки інформації. – Х.: ХУ ПС, 2008. – Вип.7(74). – С.120-123.

47. Смирнов А.А. Усовершенствование метода управления очередями в многопротокольных узлах телекоммуникационной сети / А.А.Смирнов, Е.В.Мелешко // Збірник тез та доповідей другої всеукраїнської науково-практичної конференції «Системний аналіз. Інформатика. Управління». Запоріжжя. Тези доповідей. Запоріжжя: КПУ, 2011.

48. Смирнов С.А. Метод безопасной маршрутизации метаданных в облачные антивирусные системы / А.К. Дидык, С.А. Смирнов // Информационные технологии в управлении, образовании, науке и промышленности: монография / Под редакцией профессора В.С. Пономаренко. – Х.: Видавець Рожко С.Г., 2016. – 566 с.

49. Смирнов С. А. Сравнительные исследования математических моделей технологии распространения компьютерных вирусов в информационно-телекоммуникационных сетях / Мохамад Абу Таам Гани, А. А. Смирнов,

А. В. Коваленко, С. А. Смирнов // Системи обробки інформації: зб. наук. праць. – Х.: ХУПС, 2014. – Вип. 9(125). – 105-110.

50. Смирнов С. А. Математическая модель интеллектуального узла коммутации с обслуживанием информационных пакетов различного приоритета / Мохамад Абу Таам Гани, А. А. Смирнов, Н. С. Якименко, С. А. Смирнов // Збірник наукових праць Харківського університету Повітряних Сил. – Харків: ХУПС, 2014. – Вип. 4 (41). – С. 48-52.

51. Смирнов С. А. Исследование показателей качества функционирования интеллектуальных узлов коммутации в телекоммуникационных системах и сетях / Мохамад Абу Таам Гани, А. А. Смирнов, Н. С. Якименко, С. А. Смирнов // Наука і техніка Повітряних Сил Збройних Сил України: наук. журн. –Х.: ХУПС, 2014. – № 4(17). – С. 90-95.

52. Смирнов С. А. Усовершенствованный алгоритм управления доступом к «облачным» телекоммуникационным ресурсам / Мохамад Абу Таам Гани, А. А. Смирнов, Н. С. Якименко, С. А. Смирнов // Системи обробки інформації: зб. наук. праць. – Х.: ХУПС, 2015. –Вип. 1(126). – С. 150-153.

53. Зеркалов Д. В. Охорона праці в Галузі: Загальні вимоги: навч. посіб. Київ: Основа. 2011. 551 с.

54. Про охорону праці: Закон України від 14.10.1992 р. № 2694-XII. Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/2694-12#Text

55. Сакулин В.П., Шептовицкий В.М. Безопасность труда при монтаже и эксплуатации электроустановок / В.П.Сакулин, В.М.Шептовицкий. – Л. : “Колос”, 1973. – 238 с.

56. Охорона праці. Ч. 1. Захисне заземлення: метод. вказ. до викон. розрахунків з викор. персон. ЕОМ ІВМ сумісного типу / Кіровоград. ін-т с.-г. машинобуд.; [укл. О. В. Оришака, Є. К. Солових, В. О. Оришака]. – Кіровоград: КІСМ, 1997. – 20 с. Режим доступу до ресурсу: http://dspace.kntu.kr.ua/jspui/handle/123456789/4358

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи кібербезпеки силової інфраструктури ЦОД.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 42-13 від 02.08.2021 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи кібербезпеки силової інфраструктури ЦОД.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– техніко-економічне обґрунтування доцільності прийнятого до розробки програмного забезпечення;

– аналіз умов праці;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– програмну реалізацію системи кібербезпеки силової інфраструктури ЦОД;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ архітектури IBM PC, працювати в ОС Windows XP/Vista/7/8/10 і з сумісними з цією платформою пристроями і прикладним програмним забезпечення.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows XP/Vista/7/8/10.

5.8.1 Обладнання

Комп'ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14" 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище Embarcadero RAD Studio Delphi 10.3.2 Rio Architect.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Економічні вимоги

7.1 Для ПЗ необхідно виробити функціонально-вартісний аналіз варіантів розробки.

7.2 Виконати розрахунок витрат показників економічного ефекту з урахуванням цін на 3 вересня 2021 року.

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинні бути розглянуті шкідливі і небезпечні фактори при роботі з комп’ютером.

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 122 аркуші.

10 Етапи розробки

10.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти (складання ТЗ).

10.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти.

10.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на попередній захист 10.12.2021 р.

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 20.12.2021 р.

Додаток Б

(обов'язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

другим (магістерським) рівнем вищої освіти

_____ Смірнов С.А.

Дослідження та програмна реалізація

системи кібербезпеки силової інфраструктури ЦОД

Лістинг програми

Код документу 12

Носій: CD/DVD-диск

Загальна кількість аркушів: 49

Літера: РП

Кропивницький – 2021 року

Файл TCP_IP.pas- монітор TCP/IP з'єднань мережі IoT пристроїв силової інфраструктури ЦОД

```
unit TCP_IP;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, IPHelper, IpHlpApi, Buttons;

type
  TForm2 = class(TForm)
    StaticText2: TStaticText;
    StaticText3: TStaticText;
    TCPMemo: TMemo;
    UDPMemo: TMemo;
    Timer1: TTimer;
    cbTimer: TCheckBox;
    btRTTI: TSpeedButton;
    SpeedButton1: TSpeedButton;
    edtRTTI: TEdit;
    procedure Timer1Timer(Sender: TObject);
    procedure SpeedButton1Click(Sender: TObject);
    procedure btRTTIClick(Sender: TObject);
    procedure cbRecentIPsClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
      procedure DOIpStuff;
  public
    { Public declarations }
  end;

var
  Form2: TForm2;

implementation

{$R *.dfm}

procedure TForm2.Timer1Timer(Sender: TObject);
begin
  if cbTimer.State = cbCHECKED then
 begin
   Timer1.Enabled := false;
   DoIPStuff;
   Timer1.Enabled := true;
 end;
end;

procedure TForm2.DOIpStuff;
begin

  Get_TCPTable( TCPMemo.Lines );
  Get_UDPTable( UDPMemo.Lines );

end;

procedure TForm2.SpeedButton1Click(Sender: TObject);
begin
  Speedbutton1.Enabled := false;
  DoIPStuff;
  Speedbutton1.Enabled := true;
end;

procedure TForm2.btRTTIClick(Sender: TObject);
var
```

```
  IPadr         : dword;
  Rtt, HopCount : longint;
  Res           : integer;
begin
  btRTTI.Enabled := false;
  Screen.Cursor := crHOURGLASS;
  IPadr := Str2IPAddr( edtRTTI.Text );
  Res := Get_RTTAndHopCount( IPadr, 128, RTT, HopCount );
  if Res = NO_ERROR then
    ShowMessage( ' Час запиту '
      + inttostr( rtt ) + ' ms, '
      + inttostr( HopCount )
      + ' hops to : ' + edtRTTI.Text
      )
  else
    ShowMessage( ' Відбулася помилка:' + #13
                 + ICMPErr2Str( Res ) ) ;
  btRTTI.Enabled := true;
  Screen.Cursor := crDEFAULT;

end;

procedure TForm2.cbRecentIPsClick(Sender: TObject);
begin
//edtRTTI.Text := cbRecentIPs.Items[cbRecentIPs.ItemIndex];
end;

procedure TForm2.FormCreate(Sender: TObject);
begin

  if LoadIpHlp then
  begin
      DOIpStuff;
      Timer1.Enabled := true;
  end
  else
      ShowMessage( ' Інтернет помічник DLL не є доступним, або не підтримується'
) ;
end;

end.
```

Файл Stat.pas- статистика мережі IoT пристроїв силової інфраструктури ЦОД

```
unit Stat;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, IPHelper, IpHlpApi, Buttons;

type
  TForm3 = class(TForm)
    StaticText7: TStaticText;
    TCPStatMemo: TMemo;
    StaticText5: TStaticText;
    IPStatsMemo: TMemo;
    StaticText12: TStaticText;
    ICMPInMemo: TMemo;
    ICMPOutMemo: TMemo;
    StaticText4: TStaticText;
    UDPStatsMemo: TMemo;
    Timer1: TTimer;
    cbTimer: TCheckBox;
    btRTTI: TSpeedButton;
    edtRTTI: TEdit;
    procedure Timer1Timer(Sender: TObject);
    procedure btRTTIClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
     procedure DOIpStuff;
  public
    { Public declarations }
  end;

var
  Form3: TForm3;

implementation

{$R *.dfm}

procedure TForm3.DOIpStuff;
begin

  Get_TCPStatistics( TCPStatMemo.Lines );
  Get_IPStatistics( IPStatsMemo.Lines );
  Get_UDPStatistics( UDPStatsMemo.Lines );
  Get_ICMPStats( ICMPInMemo.Lines, ICMPOutMemo.Lines );

end;

procedure TForm3.Timer1Timer(Sender: TObject);
begin
  if cbTimer.State = cbCHECKED then
 begin
   Timer1.Enabled := false;
   DoIPStuff;
   Timer1.Enabled := true;
 end;
end;

procedure TForm3.btRTTIClick(Sender: TObject);
var
  IPadr         : dword;
  Rtt, HopCount : longint;
  Res           : integer;
```

```
begin
  btRTTI.Enabled := false;
  Screen.Cursor := crHOURGLASS;
  IPadr := Str2IPAddr( edtRTTI.Text );
  Res := Get_RTTAndHopCount( IPadr, 128, RTT, HopCount );
  if Res = NO_ERROR then
    ShowMessage( ' Час запиту '
      + inttostr( rtt ) + ' ms, '
      + inttostr( HopCount )
      + ' hops to : ' + edtRTTI.Text
      )
  else
    ShowMessage( ' Помилка:' + #13
                 + ICMPErr2Str( Res ) ) ;
  btRTTI.Enabled := true;
  Screen.Cursor := crDEFAULT;

end;

procedure TForm3.FormCreate(Sender: TObject);
begin

  if LoadIpHlp then
  begin
      DOIpStuff;
      Timer1.Enabled := true;
  end
  else
      ShowMessage( 'Інтернет помічник DLL не є доступним, або не підтримується'
) ;
end;

end.
```

Основна програма

Файл Monitoring_for_IoT_DC.dpr основної програми

```
program Monitoring_for_IoT_DC;

uses
  Forms,
  Main in ' Main.pas'  {MainForm},
  About in ' About.pas'  {Form1},
  TCP_IP in ' TCP_IP.pas'  {Form2},
  Stat in ' Stat.pas'  {Form3};

{$R *.res}

begin
  Application.Initialize;
  Application.CreateForm(TMainForm, MainForm);
  Application.CreateForm(TForm1, Form1);
  Application.CreateForm(TForm2, Form2);
  Application.CreateForm(TForm3, Form3);
  Application.Run;
end.
```

Файл Main.pas основної програми

```
unit Main;

interface

// опис бібліотек

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, ComCtrls, Stat,
  ShellAPI, ShlObj, ImgList, TCP_IP, About;

//опис типів

type
  TMainForm = class(TForm)
    gbxShares: TGroupBox;
    lbxShares: TListBox;
    gbxSessions: TGroupBox;
    lvSessions: TListView;
    bvlSessions: TBevel;
    gbxFiles: TGroupBox;
    btnGetShares: TButton;
    btnCloseShares: TButton;
    btnAddShares: TButton;
    btnCloseSession: TButton;
    btnGetSessions: TButton;
    bvlTopSessions: TBevel;
    plButtonFiles: TPanel;
    btnGetFiles: TButton;
    btnCloseFile: TButton;
    bvlLeftFiles: TBevel;
    plFiles: TPanel;
    lvFiles: TListView;
    bvlTopFiles: TBevel;
    gbxTraffic: TGroupBox;
    lvTraffic: TListView;
    bvlTraffic: TBevel;
    tmrTraffic: TTimer;
    Button1: TButton;
    rgScope: TRadioGroup;
    GroupBox1: TGroupBox;
    cbUsageAll: TCheckBox;
    cbUsageConnectable: TCheckBox;
    cbUsageContainer: TCheckBox;
    GroupBox2: TGroupBox;
    cbTypeAny: TCheckBox;
    cbTypeDisk: TCheckBox;
    cbTypePrint: TCheckBox;
    NetTree: TTreeView;
    ImageList1: TImageList;
    Button2: TButton;
    Button3: TButton;
    Button4: TButton;
    function  IsNT(var Value: Boolean): Boolean;
    procedure btnGetSharesClick(Sender: TObject);
    procedure btnCloseSharesClick(Sender: TObject);
    function  SelectDirectory: String;
    procedure btnAddSharesClick(Sender: TObject);
    function  CardinalToTimeStr(Value:Cardinal):String;
    procedure btnGetSessionsClick(Sender: TObject);
    procedure btnCloseSessionClick(Sender: TObject);
    procedure btnGetFilesClick(Sender: TObject);
    procedure btnCloseFileClick(Sender: TObject);
    procedure tmrTrafficTimer(Sender: TObject);
    procedure Button1Click(Sender: TObject);
```

```
    procedure NetTreeCustomDrawItem(Sender: TCustomTreeView;
      Node: TTreeNode; State: TCustomDrawState; var DefaultDraw: Boolean);
    procedure NetTreeDblClick(Sender: TObject);
    procedure NetTreeGetImageIndex(Sender: TObject; Node: TTreeNode);
    procedure Button4Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);

//опис типів та записів


  private
    { Private declarations }
  public
    { Public declarations }
    SessionCloseKey: array [0..512] of SmallInt;
    procedure Open_Do_Close_Enum(const ParentNode: TTreeNode;
        ResScope, ResType, ResUsage: DWORD; const NetContainerToOpen:
PNetResource);
  //  function OpenEnum(const NetContainerToOpen: PNetResource;
 //        ResScope, ResType, ResUsage: DWORD): THandle;
 //   function EnumResources(const ParentNode: TTreeNode;
  //       ResScope, ResType, ResUsage: DWORD; hNetEnum: THandle): UINT;
  end;

type
 TShareInfo2 = packed record
  shi2_netname : PWChar;
  shi2_type: DWORD;
  shi2_remark :PWChar;
  shi2_permissions: DWORD;
  shi2_max_uses : DWORD;
  shi2_current_uses : DWORD;
  shi2_path : PWChar;
  shi2_passwd : PWChar;
 end;
 PShareInfo2 = ^ TShareInfo2;
 TShareInfo2Array = array [0..512] of TShareInfo2;
 PShareInfo2Array = ^ TShareInfo2Array;

type
 TShareInfo50 = packed record
  shi50_netname : array [0..12] of Char;
  shi50_type : Byte;
  shi50_flags : Word;
  shi50_remark : PChar;
  shi50_path : PChar;
  shi50_rw_password : array [0..8] of Char;
  shi50_ro_password : array [0..8] of Char;
 end;

type
 TSessionInfo502 = packed record
   Sesi502_cname: PWideChar;
   Sesi502_username: PWideChar;
   Sesi502_num_opens: DWORD;
   Sesi502_time: DWORD;
   Sesi502_idle_time: DWORD;
   Sesi502_user_flags: DWORD;
   Sesi502_cltype_name: PWideChar;
   Sesi502_transport: PWideChar;
 End;
 PSessionInfo502 = ^TSessionInfo502;
 TSessionInfo502Array = array[0..512] of TSessionInfo502;
 PSessionInfo502Array = ^TSessionInfo502Array;

type
 TSessionInfo50 = packed record
   Sesi50_cname       : PChar;
```

```
    Sesi50_username    : PChar;
    sesi50_key         : Cardinal;
    sesi50_num_conns   : Word;
    sesi50_num_opens   : Word;
    sesi50_time        : Cardinal;
    sesi50_idle_time   : Cardinal;
    sesi50_protocol    : Byte;
    pad1               : Byte;
  end;

type
  TFileInfo3 = packed record
    fi3_id          : DWORD;
    fi3_permissions : DWORD;
    fi3_num_locks   : DWORD;
    fi3_pathname    : PWChar;
    fi3_username    : PWChar;
  end;
  PFileInfo3 = ^TFileInfo3;
  TFileInfo3Array = array[0..512] of TFileInfo3;
  PFileInfo3Array = ^TFileInfo3Array;

type
  TFileInfo50 = packed record
    fi50_id          : Cardinal;
    fi50_permissions : WORD;
    fi50_num_locks   : WORD;
    fi50_pathname    : PChar;
    fi50_username    : PChar;
    fi50_sharename   : PChar;
  end;

type
   TMibIfRow = packed record
     wszName            : array[0..255] of WideChar;
     dwIndex            : DWORD;
     dwType             : DWORD;
     dwMtu              : DWORD;
     dwSpeed            : DWORD;
     dwPhysAddrLen      : DWORD;
     bPhysAddr          : array[0..7] of Byte;
     dwAdminStatus      : DWORD;
     dwOperStatus       : DWORD;
     dwLastChange       : DWORD;
     dwInOctets         : DWORD;
     dwInUcastPkts      : DWORD;
     dwInNUCastPkts     : DWORD;
     dwInDiscards       : DWORD;
     dwInErrors         : DWORD;
     dwInUnknownProtos  : DWORD;
     dwOutOctets        : DWORD;
     dwOutUCastPkts     : DWORD;
     dwOutNUCastPkts    : DWORD;
     dwOutDiscards      : DWORD;
     dwOutErrors        : DWORD;
     dwOutQLen          : DWORD;
     dwDescrLen         : DWORD;
     bDescr             : array[0..255] of Char;
   end;
   TMibIfArray = array [0..512] of TMibIfRow;
   PMibIfRow = ^TMibIfRow;
   PMibIfArray = ^TMibIfArray;

type
  TMibIfTable = packed record
    dwNumEntries     : DWORD;
    Table            : TMibIfArray;
  end;
  PMibIfTable = ^TMibIfTable;
```

```
var
NetShareEnumNT:function (     servername:PWChar;
                          level:DWORD;
                          bufptr:Pointer;
                          prefmaxlen:DWORD;
                          entriesread,
                          totalentries,
                          resume_handle:LPDWORD): DWORD; stdcall;

var
NetShareEnum:function ( pszServer   : PChar;
                        sLevel           : Cardinal;
                        pbBuffer    : Pchar;
                        cbBuffer    : Cardinal;
                        pcEntriesRead,
                        pcTotalAvail: Pointer):DWORD; stdcall;

var
NetShareDelNT:function (servername: PWideChar;
                        netname: PWideChar;
                        reserved: DWORD): LongInt; stdcall;

var
NetShareDel:function (  pszServer,
                        pszNetName:PChar;
                        usReserved:Word): DWORD; stdcall;

var
NetShareAddNT: function(servername: PWideChar;
                        level: DWORD;
                        buf: Pointer;
                        parm_err: LPDWORD): DWORD; stdcall;

var
NetShareAdd: function ( pszServer:Pchar;
                        sLevel:Cardinal;
                        pbBuffer:PChar;
                        cbBuffer:Word):DWORD; stdcall;

Var
NetSessionEnumNT:function(servername,
                        UncClientName,
                        username:PWChar;
                        level:DWORD;
                        bufptr:Pointer;
                        prefmaxlen:DWORD;
                        entriesread,
                        totalentries,
                        resume_handle:LPDWORD):DWORD; stdcall;

var
NetSessionEnum:function(pszServer:PChar;
                        sLevel: DWORD;
                        pbBuffer:Pointer;
                        cbBuffer:DWORD;
                        pcEntriesRead,
                        pcTotalAvial:Pointer):integer; stdcall;

var
NetSessionDelNT:function(ServerName,
                        UncClientName,
                        username:PWChar):DWORD; stdcall;

var
NetSessionDel:function( pszServer:PChar;
                        pszClientName: PChar;
                        sReserved: SmallInt):DWORD; stdcall;

var
```

```
NetFileEnumNT:function( servername,
                        basepath,
                        username:PWChar;
                        level:DWORD;
                        bufptr:Pointer;
                        prefmaxlen:DWORD;
                        entriesread,
                        totalentries,
                        resume_handle:LPDWORD):DWORD; stdcall;

var
NetFileEnum:function(     pszServer,
                        pszBasePath:PChar;
                        sLevel:DWORD;
                        pbBuffer:Pointer;
                        cbBuffer:DWORD;
                        pcEntriesRead,
                        pcTotalAvail:pointer):integer; stdcall;

var
NetFileClose:function(  ServerName:PWideChar;
                        FileId:DWORD):DWORD; stdcall;

var
NetFileClose2:function( pszServer:PChar;
                        ulFileId:LongWord):DWORD; stdcall;

var
GetIfTable:function(    pIfTable    : PMibIfTable;
                        pdwSize     : PULONG;
                        bOrder      : Boolean ): DWORD; stdcall;

var
  MainForm: TMainForm;

implementation

{$R *.dfm}

{ TMainForm }

//////////////////////////
//
//  Спочатку нам потрібно визначиться, під якою системою ми працюємо,
// щоб довідатися яку частину коду (для NT чи ні) використовувати в цей момент.
// Для цього напишемо невелику функцію, що і буде визначати тип системи.
//

function TMainForm.IsNT(var Value: Boolean): Boolean;
var Ver: TOSVersionInfo;
    BRes: Boolean;
begin
  Ver.dwOSVersionInfoSize := SizeOf(TOSVersionInfo);
  BRes := GetVersionEx(Ver);
  if not BRes then   //Перевірка
  begin
    Result := False; //Інформація не отримана
    Exit;            //ідемо
  end else
    Result := True;  //Інформація отримана

  case Ver.dwPlatformId of //визначаємося
    VER_PLATFORM_WIN32_NT       : Value := True;  //Windows NT тьа вище -
підходить
    VER_PLATFORM_WIN32_WINDOWS  : Value := False; //Windows 9 x-Me- підходить
    VER_PLATFORM_WIN32s         : Result := False //Windows 3.x- не підходить
  end;
end;
```

```
////////////////////////////
//
//  Одержання всіх відкритих загальних ресурсів досліджуємої мережі
//

procedure TMainForm.btnGetSharesClick(Sender: TObject);
var
  i:Integer;
  FLibHandle : THandle;
  ShareNT : PShareInfo2Array;     //<= Змінні
  entriesread,totalentries:DWORD; //<= для Windows NT
  Share : array [0..512] of TShareInfo50; //<= Змінні
  pcEntriesRead,pcTotalAvail:Word;        //<= для Windows 9 x-Me
  OS: Boolean;
begin
  lbxShares.Items.Clear;
  if not IsNT(OS) then Close; //Визначаємо тип системи

  if OS then begin  //Код для NT
    FLibHandle := LoadLibrary(' NETAPI32.DLL' ); //Завантажуємо бібліотеку
    if FLibHandle = 0 then Exit;
    //Зв' язуємо функцію
    @NetShareEnumNT := GetProcAddress(FLibHandle,' NetShareEnum' );
    if not Assigned(NetShareEnumNT) then //Перевірка
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    ShareNT := nil; //Очищаємо покажчик на масив структур
    //Виклик функції
    if NetShareEnumNT(nil,2,@ShareNT,DWORD(-1),
                      @entriesread,@totalentries,nil) <> 0 then
    begin  //Якщо виклик невдалий вивантажуємо бібліотеку
      FreeLibrary(FLibHandle);
      Exit;
    end;
    if entriesread > 0 then //Обробка результатів
    for i:= 0 to entriesread- 1 do
      lbxShares.Items.Add(String(ShareNT^[i].shi2_netname));
  end else begin //Код для 9 x-me
    FLibHandle := LoadLibrary(' SVRAPI.DLL' ); //Завантажуємо бібліотеку
    if FLibHandle = 0 then Exit;
    //Зв' язуємо функцію
    @NetShareEnum := GetProcAddress(FLibHandle,' NetShareEnum' );
    if not Assigned(NetShareEnum) then //Перевірка
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    //Виклик функції
    if NetShareEnum(nil,50,@Share,SizeOf(Share),
                  @pcEntriesRead,@pcTotalAvail)<> 0 then
    begin  //Якщо виклик невдалий вивантажуємо бібліотеку
      FreeLibrary(FLibHandle);
      Exit;
    end;
    if pcEntriesRead > 0 then //Обробка результатів
    for i:= 0 to pcEntriesRead- 1 do
      lbxShares.Items.Add(String(Share[i].shi50_netname));
  end;
  FreeLibrary(FLibHandle); //Не забуваємо вивантажити бібліотеку
end;

////////////////////////////
//
//  Закриття загального ресурсу
//

procedure TMainForm.btnCloseSharesClick(Sender: TObject);
```

```
var
  OS:Boolean;
  FLibHandle : THandle;
  Name9x:array [0..12] of Char;
  NameNT:PWChar;
  i:Integer;
  ShareName: String;
begin
  if not IsNT(OS) then Close; //Визначаємо тип системи

  if lbxShares.Items.Count = 0 then Exit;
  for i:= 0 to lbxShares.Items.Count-1 do
    if lbxShares.Selected[i] then Break;  //Шукаємо обраний елемент
  if not lbxShares.Selected[i] then Exit; //Якщо не знайдений ідемо
  ShareName := lbxShares.Items.Strings[i];

  if OS then begin //Код для NT
    FLibHandle := LoadLibrary(' NETAPI32.DLL' );
    if FLibHandle = 0 then Exit;
    @NetShareDelNT := GetProcAddress(FLibHandle,' NetShareDel' );
    if not Assigned(NetShareDelNT) then //Перевірка
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    i:= SizeOf(WideChar)*256;
    GetMem(NameNT,i);  //Виділяємо пам' ять під змінну
    StringToWideChar(ShareName,NameNT,i); //Перетворимо в PWideChar
    NetShareDelNT(nil,NameNT,0);   //Видаляємо ресурс
    FreeMem(NameNT);  //Звільняємо пам' ять
  end else begin  //Код для 9 х-ме
    FLibHandle := LoadLibrary(' SVRAPI.DLL' );
    if FLibHandle = 0 then Exit;
    @NetShareDel := GetProcAddress(FLibHandle,' NetShareDel' );
    if not Assigned(NetShareDel) then //Перевірка
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    FillChar(Name9x, SizeOf(Name9x), #0); //Очищаємо масив
    move(ShareName[1],Name9x[0],Length(ShareName)); //Заповнюємо масив
    NetShareDel(nil,@Name9x,0); //Видаляємо ресурс
  end;
  FreeLibrary(FLibHandle);
end;

//////////////////////////
//
//  Показ діалогу вибору директорії
//

function TMainForm.SelectDirectory: String;
var
  lpItemID : PItemIDList;
  BrowseInfo : TBrowseInfo;
  DisplayName : array[0..MAX_PATH] of Char;
  TempPath : array[0..MAX_PATH] of Char;
begin
  FillChar(BrowseInfo, sizeof(TBrowseInfo), #0);
  BrowseInfo.hwndOwner := Handle;
  BrowseInfo.pszDisplayName := @DisplayName;
  BrowseInfo.lpszTitle := ' Specify a directory' ;
  BrowseInfo.ulFlags := BIF_RETURNONLYFSDIRS;
  lpItemID := SHBrowseForFolder(BrowseInfo);
  if Assigned(lpItemId) then begin
    SHGetPathFromIDList(lpItemID, TempPath);
    GlobalFreePtr(lpItemID);
  end else Result := ' ' ;
  Result := String(TempPath);
```

```
end;

//////////////////////////
//
//  Додавання загального ресурсу
//

procedure TMainForm.btnAddSharesClick(Sender: TObject);
const
  STYPE_DISKTREE = 0;
  ACCESS_ALL =  258;
  SHI50F_FULL = 258;
var
  FLibHandle : THandle;
  Share9x : TShareInfo50;
  ShareNT : TShareInfo2;
  TmpDir, TmpName: String;
  TmpDirNT, TmpNameNT: PWChar;
  OS: Boolean;
  TmpLength: Integer;
begin
  TmpDir := SelectDirectory; //Визначаємо шлях до наступного ресурсу
  TmpName := InputBox(' Share name' ,' Enter name' ,' Test' ); //Визначаємо ім'
я під яким він буде видний у мережі
  if TmpDir = ' '  then Exit;

  if not IsNT(OS) then Close; //З' ясовуємо тип системи

  if OS then begin //Код для NT
    FLibHandle := LoadLibrary(' NETAPI32.DLL' );
    if FLibHandle = 0 then Exit;
    @NetShareAddNT := GetProcAddress(FLibHandle,' NetShareAdd' );
    if not Assigned(NetShareAddNT) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    TmpLength := SizeOF(WideChar)*256; //Визначаємо необхідний розмір

    GetMem(TmpNameNT, TmpLength); //Конвертуємо в PWChar
    StringToWideChar(TmpName, TmpNameNT, TmpLength);
    ShareNT.shi2_netname := TmpNameNT; //Ім' я

    ShareNT.shi2_type := STYPE_DISKTREE; //Тип ресурсу
    ShareNT.shi2_remark := ' ' ; //Коментар
    ShareNT.shi2_permissions := ACCESS_ALL; //Доступ
    ShareNT.shi2_max_uses := DWORD(-1); // Кіл-ть максим. підключ.
    ShareNT.shi2_current_uses := 0; // Кіл-ть поточних підкл.

    GetMem(TmpDirNT, TmpLength);
    StringToWideChar(TmpDir, TmpDirNT, TmpLength);
    ShareNT.shi2_path := TmpDirNT; //Шлях до ресурсу

    ShareNT.shi2_passwd := ' ' ; //Пароль

    NetShareAddNT(nil,2,@ShareNT, nil); //Додаємо ресурс
    FreeMem (TmpNameNT); //звільняємо пам' ять
    FreeMem (TmpDirNT);
  end else begin
    FLibHandle := LoadLibrary(' SVRAPI.DLL' );
    if FLibHandle = 0 then Exit;
    @NetShareAdd := GetProcAddress(FLibHandle,' NetShareAdd' );
    if not Assigned(NetShareAdd) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    FillChar(Share9x.shi50_netname, SizeOf(Share9x.shi50_netname), #0);
    move(TmpName[1],Share9x.shi50_netname[0],Length(TmpName)); //Ім' я
```

```
    Share9x.shi50_type := STYPE_DISKTREE; //Тип ресурсу
    Share9x.shi50_flags := SHI50F_FULL; //Доступ
    FillChar(Share9x.shi50_remark,
      SizeOf(Share9x.shi50_remark), #0); //Коментар
    FillChar(Share9x.shi50_path,
      SizeOf(Share9x.shi50_path), #0);
    Share9x.shi50_path := PAnsiChar(TmpDir); //Шлях до ресурсу
    FillChar(Share9x.shi50_rw_password,
      SizeOf(Share9x.shi50_rw_password), #0); //Пароль повного доступу
    FillChar(Share9x.shi50_ro_password,
      SizeOf(Share9x.shi50_ro_password), #0); //Пароль для читання
    NetShareAdd(nil,50,@Share9x,SizeOf(Share9x));
  end;
  FreeLibrary(FLibHandle);
end;

////////////////////////////
//
//  Помітьте що активний і неактивний час сесій буде даватися нам
//  у вигляді кіл-ті секунд (тип Cardinal). Напишемо невелику
//  функцію, задача якої буде перетворювати Кіл-ть секунд у більше
//  звичну форму відображення.
//

function TMainForm.CardinalToTimeStr(Value: Cardinal): String;
var d,h,m,s: Real;
begin
    d:=0;
    h:=0;
    m:=0;
    s:=Value;
    if s > 59 then begin
        m:=int(s / 60);
        s:= s-s-(m*60);
    end;
    if m > 59 then begin
        h:=int(m/60);
        m:= m-m-(h*60);
    end;
    if h > 23 then begin
        d:=int(h/24);
        h:= h-h-(d*24);
    end;
    Result:=' ' ;
    if (d>0) then Result:=Result+floattostr(d)+'  d. ' ;
    if (h<9) then Result:=Result+' 0' +floattostr(h)+' :'  else
Result:=Result+floattostr(h)+' :' ;
    if (m<9) then Result:=Result+' 0' +floattostr(m)+' :'  else
Result:=Result+floattostr(m)+' :' ;
    if (s<9) then Result:=Result+' 0' +floattostr(s)     else
Result:=Result+floattostr(s);
end;

////////////////////////////
//
//  Одержання списку сесій мережі IoT пристроїв силової інфраструктури ЦОД
//

procedure TMainForm.btnGetSessionsClick(Sender: TObject);
var
  OS: Boolean;
  FLibHandle : THandle;
  SessionInfo50: array [0..512] of TSessionInfo50;
  SessionInfo502 : PSessionInfo502Array;
  TotalEntries,EntriesReadNT: DWORD;
  EntriesRead,TotalAvial: Word;
  i:integer;
begin
  lvSessions.Items.Clear;
```

```
  if not IsNT(OS) then Close; //З' ясовуємо тип системи

  if OS then begin  //Код для NT
    FLibHandle := LoadLibrary(' NETAPI32.DLL' );
    if FLibHandle = 0 then Exit;
    @NetSessionEnumNT := GetProcAddress(FLibHandle, ' NetSessionEnum' );
    if not Assigned(NetSessionEnumNT) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    SessionInfo502 := nil;
    if NetSessionEnumNT(nil,nil,nil,502,@SessionInfo502,DWORD(-
1),@entriesreadNT, @totalentries, nil)=0 then
    for i:=0 to EntriesReadNT-1 do
    begin
      with lvSessions.Items.Add do  //Заповнення даними зі структури
      begin
        Caption := string(SessionInfo502^[i].sesi502_cname); //Ім' я комп' ютера
        SubItems.Add(SessionInfo502^[i].sesi502_username);   //Ім' я користувача
        SubItems.Add(IntToStr(SessionInfo502^[i].sesi502_num_opens));
//Відкритих ресурсів
        SubItems.Add(CardinalToTimeStr(SessionInfo502^[i].Sesi502_Time)); //Час
активний
        SubItems.Add(CardinalToTimeStr(SessionInfo502^[i].sesi502_idle_time));
//Час не активний
      end;
    end;
  end else begin //Код для Windows 9 x-Me
   FLibHandle := LoadLibrary(' SVRAPI.DLL' );
   if FLibHandle = 0 then Exit;
   @NetSessionEnum := GetProcAddress(FLibHandle, ' NetSessionEnum' );
    if not Assigned(NetSessionEnum) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
   if NetSessionEnum
(nil,50,@SessionInfo50,SizeOf(SessionInfo50),@EntriesRead,@TotalAvial) = 0 then
   for i:=0 to EntriesRead-1 do
   begin
      with lvSessions.Items.Add do  //Заповнення даними зі структури
      begin
        Caption := string(SessionInfo50[i].Sesi50_cname); //Ім' я комп' ютера
мережі IoT пристроїв силової інфраструктури ЦОД
        SubItems.Add(SessionInfo50[i].Sesi50_username);   //Ім' я користувача
        SubItems.Add(IntToStr(SessionInfo50[i].sesi50_num_opens)); //Відкритих
ресурсів
        SubItems.Add(CardinalToTimeStr(SessionInfo50[i].Sesi50_Time)); //Час
активний
        SubItems.Add(CardinalToTimeStr(SessionInfo50[i].sesi50_idle_time));
//Час не активний
        SessionCloseKey[i]:= SessionInfo50[i].sesi50_key; //Унікальний
ідентифікатор для закриття
      end;
   end;
 end;
 FreeLibrary(FLibHandle);
end;

////////////////////////////
//
//  Завершення обраної сесії
//

procedure TMainForm.btnCloseSessionClick(Sender: TObject);
var
  OS: Boolean;
```

```
  FLibHandle : THandle;
  CNameNT: PWideChar;
  CName9x: PAnsiChar;
  Key:SmallInt;
  i: Integer;
begin
  if not IsNT(OS) then Close; //З' ясовуємо тип системи

  if not Assigned(lvSessions.Selected) then Exit;
  i:= lvSessions.Selected.Index; //Визначаємо номер обраної сесії

  if OS then begin
    FLibHandle := LoadLibrary(' NETAPI32.DLL' );
    if FLibHandle = 0 then Exit;
    @NetSessionDelNT := GetProcAddress(FLibHandle, ' NetSessionDel' );
    if not Assigned(NetSessionDelNT) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    //Перетворимо дані в необхідний вид
    CNameNT := PWChar(WideString(' \\' +lvSessions.Items.Item[i].Caption));
    NetSessionDelNT(nil,CNameNT,nil);
  end else begin
    FLibHandle := LoadLibrary(' SVRAPI.DLL' );
    if FLibHandle = 0 then Exit;
    @NetSessionDel := GetProcAddress(FLibHandle, ' NetSessionDel' );
    if not Assigned(NetSessionDel) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    //Перетворимо дані в необхідний вид
    CName9x := PAnsiChar(lvSessions.Items.Item[i].Caption);
    key := SessionCloseKey[i]; //Беремо ключ із масиву
    NetSessionDel(nil,CName9x,Key);
  end;
  FreeLibrary(FLibHandle);
end;

////////////////////////////
//
//  Одержання списку відкритих файлів мережі IoT пристроїв силової
інфраструктури ЦОД
//

procedure TMainForm.btnGetFilesClick(Sender: TObject);
var
  OS: Boolean;
  FLibHandle : THandle;
  FileInfoNT: PFileInfo3Array;
  FileInfo9x: array [0..512] of TFileInfo50;
  TotalEntries,EntriesReadNT: DWORD;
  EntriesRead,TotalAvial: Word;
  i:integer;
begin
  lvfiles.Items.Clear;

  if not IsNT(OS) then Close; //З' ясовуємо тип системи

  if OS then begin  //Код для NT
    FLibHandle := LoadLibrary(' NETAPI32.DLL' );
    if FLibHandle = 0 then Exit;
    @NetFileEnumNT := GetProcAddress(FLibHandle, ' NetFileEnum' );
    if not Assigned(NetFileEnumNT) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
```

```
    FileInfoNT := nil;
    if NetFileEnumNT(nil,nil,nil,3,@FileInfoNT,DWORD(-1),@entriesreadNT,
@totalentries, nil)=0 then
    for i:=0 to EntriesReadNT-1 do
    begin
      with lvFiles.Items.Add do  //Заповнення даними зі структури
      begin
        Caption := string(IntToStr(FileInfoNT^[i].fi3_id)); //Ідентифікатор
        SubItems.Add(FileInfoNT^[i].fi3_pathname);   //Шлях до файлу
        SubItems.Add(FileInfoNT^[i].fi3_username);   //Ім' я користувача
      end;
    end;
  end else begin //Код для Windows 9 x-Me
   FLibHandle := LoadLibrary(' SVRAPI.DLL' );
   if FLibHandle = 0 then Exit;
   @NetFileEnum := GetProcAddress(FLibHandle, ' NetFileEnum' );
    if not Assigned(NetFileEnum) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
   if NetFileEnum (nil,
nil,50,@FileInfo9x,SizeOf(FileInfo9x),@EntriesRead,@TotalAvial)= 0 then
   for i:=0 to EntriesRead-1 do
   begin
      with lvFiles.Items.Add do  //Заповнення даними зі структури
      begin
        Caption := string(IntToStr(FileInfo9x[i].fi50_id)); //Ідентифікатор
        SubItems.Add(FileInfo9x[i].fi50_pathname); //Шлях до файлу
        SubItems.Add(FileInfo9x[i].fi50_username); //Ім' я користувача
      end;
   end;
 end;
 FreeLibrary(FLibHandle);
end;

//////////////////////////
//
//  Закриття файлу
//

procedure TMainForm.btnCloseFileClick(Sender: TObject);
var
  OS: Boolean;
  FLibHandle : THandle;
  i: Integer;
begin
  if not IsNT(OS) then Close; //З' ясовуємо тип системи

  if not Assigned(lvFiles.Selected) then Exit;
  i:= lvFiles.Selected.Index; //Визначаємо номер обраного файлу

  if OS then begin //Код для NT
    FLibHandle := LoadLibrary(' NETAPI32.DLL' );
    if FLibHandle = 0 then Exit;
    @NetFileClose := GetProcAddress(FLibHandle, ' NetFileClose' );
    if not Assigned(NetFileClose) then
    begin
      FreeLibrary(FLibHandle);
      Close;
    end;
    NetFileClose(nil,StrToInt(lvFiles.Items.Item[i].Caption));//Закриваємо файл
  end else begin //Код для Windows 9 x-Me
    FLibHandle := LoadLibrary(' SVRAPI.DLL' );
    if FLibHandle = 0 then Exit;
    @NetFileClose2 := GetProcAddress(FLibHandle, ' NetFileClose2' );
    if not Assigned(NetFileClose2) then
    begin
      FreeLibrary(FLibHandle);
```

```
      Close;
    end;
    NetFileClose2(nil,StrToInt(lvFiles.Items.Item[i].Caption));
  end;
  FreeLibrary(FLibHandle);
end;

////////////////////////////
//
//  Визначаємо вхідний / вихідний трафік мережі IoT пристроїв силової
інфраструктури ЦОД
//

procedure TMainForm.tmrTrafficTimer(Sender: TObject);
  // Допоміжна функція, що перетворить MAC адресу до "нормального" виду
  //Визначаємо спеціальний тип, щоб можна було передати у функцію масив
  type TMAC = array [0..7] of Byte;
  //Як перше значення масив, друге значення, розмір даних у масиві
  function GetMAC(Value: TMAC; Length: DWORD): String;
  var
    i: Integer;
  begin
    if Length = 0 then Result := '  00-00-00'  else
    begin
      Result := ' ' ;
      for i:= 0 to Length-2 do
        Result := Result + IntToHex(Value[i],2)+' -' ;
      Result := Result + IntToHex(Value[ Length-1],2);
    end;
  end;

//Сама процедура
var
  FLibHandle : THandle;
  Table: TMibIfTable;
  i : integer;
  Size   : integer;
begin
  tmrTraffic.Enabled := false; //Припиняємо таймер
  lvTraffic.Items.BeginUpdate;
  lvTraffic.Items.Clear;  //Очищаємо список
  FLibHandle := LoadLibrary(' IPHLPAPI.DLL' ); //Завантажуємо бібліотеку
  if FLibHandle = 0 then Exit;
  @GetIfTable := GetProcAddress(FLibHandle, ' GetIfTable' );
  if not Assigned(GetIfTable) then
  begin
    FreeLibrary(FLibHandle);
    Close;
  end;

  Size := SizeOf(Table);
  if GetIfTable(@Table, @Size, false ) = 0 then //Виконуємо функцію
    for i:= 0 to Table.dwNumEntries-1 do begin
      with lvTraffic.Items.Add do begin //Виводимо результати
        Caption := String(Table.Table[i].bDescr); //Найменування інтерфейсу
        SubItems.Add(GetMAC(TMAC(Table.Table[i].bPhysAddr),
          Table.Table[i].dwPhysAddrLen)); //MAC адреса
        SubItems.Add(IntToStr(Table.Table[i].dwInOctets)); //Усього прийнято
байт з мережі IoT пристроїв силової інфраструктури ЦОД
        SubItems.Add(IntToStr(Table.Table[i].dwOutOctets)); //Усього відправлено
байт у досліджуєму мережу для виявлення деструктивнимх дій дестабілізуючого
трафіку(IoT_DC)
      end;
    end;
  lvTraffic.Items.EndUpdate;
  FreeLibrary(FLibHandle);
  tmrTraffic.Enabled := true; //Не забуваємо активувати таймер
end;
```

```
 function OpenEnum(const NetContainerToOpen: PNetResource; ResScope, ResType,
ResUsage: DWORD): THandle;
var
 hNetEnum: THandle;
begin
 Result:=0;
 if (NO_ERROR<>WNetOpenEnum(ResScope, ResType, ResUsage,
                                  NetContainerToOpen, hNetEnum))
 then ShowMessage(‘ Помилка!’ )
 else Result:=hNetEnum;
end;

function EnumResources(const ParentNode: TTreeNode;
ResScope, ResType, ResUsage: DWORD; hNetEnum: THandle): UINT;
 function ShowResource(const ParentNode: TTreeNode; Res: TNetResource):
TTreeNode;
  begin
   Result:=MainForm.NetTree.Items.AddChild(ParentNode,
string(Res.lpRemoteName));
  end;

const
 RESOURCE_BUF_ENTRIES = 2000;

var
 ResourceBuffer: array[1..RESOURCE_BUF_ENTRIES] of TNetResource;
 i, ResourceBuf, EntriesToGet: dword;
 NewNode: TTreeNode;
begin
 Result:=0;
 while true do
  begin
   ResourceBuf:=sizeof(ResourceBuffer);
   EntriesToGet:=RESOURCE_BUF_ENTRIES;
   if (NO_ERROR<>WNetEnumResource(hNetEnum, EntriesToGet,
                               @ResourceBuffer, ResourceBuf))
   then
    begin
     case GetLastError() of
      NO_ERROR:  // проход буферу без перемикання
      Break;
      ERROR_NO_MORE_ITEMS:
          // Повертає 0 у тому випадку, коли останов
          // RESOURCE_BUF_ENTRIES дані на попередньому виклику, щоб
          // WNetEnumResource, та були точно
          // RESOURCE_BUF_ENTRIES дані в запису на момент
          // попереднього виклику
      Exit;
      else ShowMessage(Помилка!’ );
      Result:=1;
      Exit;
     end;
    end;
    for i:=1 to EntriesToGet do
     begin
      NewNode:=ShowResource(ParentNode, ResourceBuffer[i]);
      if (ResourceBuffer[i].dwUsage and RESOURCEUSAGE_CONTAINER)<>0
      then MainForm.Open_Do_Close_Enum(NewNode, ResScope, ResType, ResUsage,
@ResourceBuffer[i]);
      Application.ProcessMessages;
     end;
  end;
end;

procedure TMainForm.Open_Do_Close_Enum(const ParentNode: TTreeNode; ResScope,
ResType, ResUsage: DWORD; const NetContainerToOpen: PNetResource);
var
 hNetEnum: THandle;
```

```
begin
 hNetEnum:=OpenEnum(NetContainerToOpen, ResScope, ResType, ResUsage);
 if (hNetEnum=0)
 then Exit;
 EnumResources(ParentNode, ResScope, ResType, ResUsage, hNetEnum);
 if (NO_ERROR<>WNetCloseEnum(hNetEnum))
 then ShowMessage(' WNetCloseEnum Помилка' );
end;

procedure TMainForm.Button1Click(Sender: TObject);
var
 ResScope, ResType, ResUsage: dword;
begin
 Button1.Caption:=' Пошук мережних ресурсів. Чекайте...' ;
 Button1.Enabled:=false;
 //
 NetTree.Items.Clear;
 case rgScope.ItemIndex of
  1: ResScope:=RESOURCE_GLOBALNET;
  2: ResScope:=RESOURCE_REMEMBERED;
  else ResScope:=RESOURCE_CONNECTED;
 end;
 ResType:=0;
 if cbTypeAny.Checked
 then ResType:=ResType or RESOURCETYPE_ANY;
 if cbTypeDisk.Checked
 then ResType:=ResType or RESOURCETYPE_DISK;
 if cbTypePrint.Checked
 then ResType:=ResType or RESOURCETYPE_PRINT;
 ResUsage:=0;
 if cbUsageConnectable.Checked
 then ResUsage:=ResUsage or RESOURCEUSAGE_CONNECTABLE;
 if cbUsageContainer.Checked
 then ResUsage:=ResUsage or RESOURCEUSAGE_CONTAINER;
 Open_Do_Close_Enum(NetTree.Items.Add(nil, ' Network Resources' ),
                                ResScope, ResType, ResUsage, nil);
 //
 Button1.Caption:=' Обновити список ресурсів' ;
 Button1.Enabled:=true;

end;

procedure TMainForm.NetTreeCustomDrawItem(Sender: TCustomTreeView;
  Node: TTreeNode; State: TCustomDrawState; var DefaultDraw: Boolean);
begin
if cdsSelected in State
 then Sender.Canvas.Font.Style:=Sender.Canvas.Font.Style+[fsUnderline];
end;

procedure TMainForm.NetTreeDblClick(Sender: TObject);
begin
ShellExecute(0,' open' ,PChar(NetTree.Selected.Text),' ' ,' ' ,SW_SHOW);
end;

procedure TMainForm.NetTreeGetImageIndex(Sender: TObject; Node: TTreeNode);
begin
 if Node.HasChildren
 then Node.ImageIndex:=1
 else Node.ImageIndex:=0;
end;

procedure TMainForm.Button4Click(Sender: TObject);
begin
Form1.Show;
end;

procedure TMainForm.Button2Click(Sender: TObject);
begin
Form2.Show;
```

```
end;

procedure TMainForm.Button3Click(Sender: TObject);
begin
Form3.Show;
end;

end.
```

Файл IPHLPAPI.pas - обробка API функцій

```
unit IPHLPAPI;

interface
uses
  Windows, winsock;

const
  VERSION       = ' 1.5' ;

//------- Заголовок з Microsoft IPTYPES.H-H-----------------

const
  ANY_SIZE      = 1;
  MAX_ADAPTER_DESCRIPTION_LENGTH = 128; // arb.
  MAX_ADAPTER_NAME_LENGTH = 256; // змінна
  MAX_ADAPTER_ADDRESS_LENGTH = 8; // змінна
  DEFAULT_MINIMUM_ENTITIES = 32; // змінна
  MAX_HOSTNAME_LEN = 128; // змінна
  MAX_DOMAIN_NAME_LEN = 128; // змінна
  MAX_SCOPE_ID_LEN = 256; // змінна

 // Вузлові типи ( NETBIOS)
  BROADCAST_NODETYPE = 1;
  PEER_TO_PEER_NODETYPE = 2;
  MIXED_NODETYPE = 4;
  HYBRID_NODETYPE = 8;

  NETBIOSTypes  : array[0..8] of string[20] =
    ( ' Невизначений' , ' Передача' , ' Рівень до рівня' , ' ' , '
Змішаний' , ' ' , ' ' , ' ' , ' Гібрід'
      );

 // Типи адаптеру
{ v1.4-> 1.5
  IF_OTHER_ADAPTERTYPE = 0;
  IF_ETHERNET_ADAPTERTYPE = 1;
  IF_TOKEN_RING_ADAPTERTYPE = 2;
  IF_FDDI_ADAPTERTYPE = 3;
  IF_PPP_ADAPTERTYPE = 4;
  IF_LOOPBACK_ADAPTERTYPE = 5;
  IF_SLIP_ADAPTERTYPE = 6;

Знайдено у ipifcons.h :
#define MIB_IF_TYPE_OTHER               1
#define MIB_IF_TYPE_ETHERNET            6
#define MIB_IF_TYPE_TOKENRING           9
#define MIB_IF_TYPE_FDDI                15
#define MIB_IF_TYPE_PPP                 23
#define MIB_IF_TYPE_LOOPBACK            24
#define MIB_IF_TYPE_SLIP                28
}
  IF_OTHER_ADAPTERTYPE = 1;
  IF_ETHERNET_ADAPTERTYPE = 6;
  IF_TOKEN_RING_ADAPTERTYPE = 9;
  IF_FDDI_ADAPTERTYPE = 15;
  IF_PPP_ADAPTERTYPE = 23;
  IF_LOOPBACK_ADAPTERTYPE = 24;
  IF_SLIP_ADAPTERTYPE = 28;

//  AdaptTypes    : array[0..6] of string[10] =
//    ( ' інший' , ' ethernet' , ' tokenring' , ' FDDI' , ' PPP' , '
loopback' , ' SLIP'  );
  AdaptTypes    : array[1..28] of string[10] =
```

```
          ( ' інший' , ' ' , ' ' , ' ' , ' ' , ' ethernet' , ' ' , ' ' , '
tokenring' ,
           ' ' , ' ' , ' ' , ' ' , ' ' , ' FDDI' , ' ' , ' ' , ' ' , ' ' , ' ' ,
' ' , ' ' , ' PPP' ,
           ' loopback' , ' ' , ' ' , ' ' , ' SLIP'  );
      // Кінець змін в типі адаптерів

      //-------для інших MS заготовочних файлів------------------

        MAX_INTERFACE_NAME_LEN = 256; { mrapi.h }
        MAXLEN_PHYSADDR = 8; { iprtrmib.h }
        MAXLEN_IFDESCR = 256; {-"--     }

      //-----------------------------------

      type
        TMacAddress = array[1..MAX_ADAPTER_ADDRESS_LENGTH] of byte;

      //---IP адресні структури-------------------------

        PTIP_ADDRESS_STRING = ^TIP_ADDRESS_STRING;
        TIP_ADDRESS_STRING = array[0..15] of char; //  IP рядок
        //
        PTIP_ADDR_STRING = ^TIP_ADDR_STRING;
        TIP_ADDR_STRING = packed record // для використання у зв' язних списках
          Next: PTIP_ADDR_STRING;
          IpAddress: TIP_ADDRESS_STRING;
          IpMask: TIP_ADDRESS_STRING;
          Context: DWORD;
        end;

      //-----Fixed Info структура---------------------

        PTFixedInfo = ^TFixedInfo;
        TFixedInfo = packed record
          HostName: array[1..MAX_HOSTNAME_LEN + 4] of char;    // дані
          DomainName: array[1..MAX_DOMAIN_NAME_LEN + 4] of char;   // дані
          CurrentDNSServer: PTIP_ADDR_STRING;
          DNSServerList: TIP_ADDR_STRING;
          NodeType: UINT;
          ScopeID: array[1..MAX_SCOPE_ID_LEN + 4] of char;   // дані
          EnableRouting: UINT;
          EnableProxy: UINT;
          EnableDNS: UINT;
        end;

      //-----структура мережного інтерфейсу мережі IoT пристроїв силової
інфраструктури ЦОД-----------------------

      ////////////////////////
      //
//
      // Наступне є діючими станами для WAN да LAN інтерфейсів. //
      // Порядок станів створений для визначення. Для      //
      // стану >= CONNECTED можливо передавати дані зразу. Стан >= DISCONNECTED
//
      // може передавати деякі дані. Стан < DISCONNECTED може   //
      // не передавати дані.
//
      // карта з поміткою UNREACHABLE якщо DIM викликає InterfaceUnreachable для
//
      // причин. Крім невдачі з'єднання.                                  //
      //
//
      // NON_OPERATIONAL- Перевірка для LAN інтерфейсу. Позначає карту що не
працює       //
      //                       або не з'єднується з картою.        //
      // UNREACHABLE- Перевірка WAN інтерфейсів . Позначає, що віддалений сайт
//
```

```
      //                      не з'єднується за потрібний час.
//
      // DISCONNECTED- Перевірка WAN інтерфейсів . Позначає, що віддалений сайт
//
      //                      не з'єднується.
      //
      // CONNECTING- Перевірка WAN інтерфейсів . Означає спробу з'єднання     //
      //                      з сайтом, якого немає.              //
      // CONNECTED- Перевірка WAN інтерфейсів . Позначає, що віддалений сайт
//
      //                      з'єднується.
//
      // OPERATIONAL- Перевірка LAN Interfaces. Позначає карту підключену //
      //                      в праці.                                    //
      //
//
      // Усі дії користувачів записуються до  MIB-II значення    //
      // можуть бути використовані                                        //
      //
//
      ////////////////////////

      const
      // дані додані до ipifcons.h
        IF_OPER_STATUS_NON_OPERATIONAL = 0 ;
        IF_OPER_STATUS_UNREACHABLE = 1 ;
        IF_OPER_STATUS_DISCONNECTED = 2 ;
        IF_OPER_STATUS_CONNECTING = 3 ;
        IF_OPER_STATUS_CONNECTED = 4 ;
        IF_OPER_STATUS_OPERATIONAL = 5 ;

        MIB_IF_TYPE_OTHER = 1 ;
        MIB_IF_TYPE_ETHERNET = 6 ;
        MIB_IF_TYPE_TOKENRING = 9 ;
        MIB_IF_TYPE_FDDI = 15 ;
        MIB_IF_TYPE_PPP = 23 ;
        MIB_IF_TYPE_LOOPBACK = 24 ;
        MIB_IF_TYPE_SLIP = 28 ;

        MIB_IF_ADMIN_STATUS_UP = 1 ;
        MIB_IF_ADMIN_STATUS_DOWN = 2 ;
        MIB_IF_ADMIN_STATUS_TESTING = 3 ;

        MIB_IF_OPER_STATUS_NON_OPERATIONAL = 0 ;
        MIB_IF_OPER_STATUS_UNREACHABLE = 1 ;
        MIB_IF_OPER_STATUS_DISCONNECTED = 2 ;
        MIB_IF_OPER_STATUS_CONNECTING = 3 ;
        MIB_IF_OPER_STATUS_CONNECTED = 4 ;
        MIB_IF_OPER_STATUS_OPERATIONAL = 5 ;

      type
        PTMibIfRow = ^TMibIfRow;
        TMibIfRow = packed record
          wszName: array[1..MAX_INTERFACE_NAME_LEN] of WCHAR;
          dwIndex: DWORD;
          dwType: DWORD;       // дивись MIB_IF_TYPE
          dwMTU: DWORD;
          dwSpeed: DWORD;
          dwPhysAddrLen: DWORD;
          bPhysAddr: array[1..MAXLEN_PHYSADDR] of byte;
          dwAdminStatus: DWORD;    // дивись MIB_IF_ADMIN_STATUS
          dwOperStatus: DWORD;     // дивись MIB_IF_OPER_STATUS
          dwLastChange: DWORD;
          dwInOctets: DWORD;
          dwInUcastPkts: DWORD;
          dwInNUCastPkts: DWORD;
          dwInDiscards: DWORD;
          dwInErrors: DWORD;
          dwInUnknownProtos: DWORD;
```

```
          dwOutOctets: DWORD;
          dwOutUCastPkts: DWORD;
          dwOutNUCastPkts: DWORD;
          dwOutDiscards: DWORD;
          dwOutErrors: DWORD;
          dwOutQLen: DWORD;
          dwDescrLen: DWORD;
          bDescr: array[1..MAXLEN_IFDESCR] of char; //byte;
        end;

       //
        PTMibIfTable = ^TMIBIfTable;
        TMibIfTable = packed record
          dwNumEntries: DWORD;
          Table: array[0..ANY_SIZE- 1] of TMibIfRow;
        end;

      //---ADAPTER INFO структура-----------------------

        PTIP_ADAPTER_INFO = ^TIP_ADAPTER_INFO;
        TIP_ADAPTER_INFO = packed record
          Next: PTIP_ADAPTER_INFO;
          ComboIndex: DWORD;
          AdapterName: array[1..MAX_ADAPTER_NAME_LENGTH + 4] of char;       //
дані
          Description: array[1..MAX_ADAPTER_DESCRIPTION_LENGTH + 4] of char;
// дані
          AddressLength: UINT;
          Address: array[1..MAX_ADAPTER_ADDRESS_LENGTH] of byte;      // дані
          Index: DWORD;
          aType: UINT;
          DHCPEnabled: UINT;
          CurrentIPAddress: PTIP_ADDR_STRING;
          IPAddressList: TIP_ADDR_STRING;
          GatewayList: TIP_ADDR_STRING;
          DHCPServer: TIP_ADDR_STRING;
          HaveWINS: BOOL;
          PrimaryWINSServer: TIP_ADDR_STRING;
          SecondaryWINSServer: TIP_ADDR_STRING;
          LeaseObtained: LongInt ; // UNIX час, секунди з 1970
          LeaseExpires: LongInt;   // UNIX час, секунди з 1970
          SpareStuff: array [1..200] of char ;   // дані- простір для списку IP
адрес мережі IoT пристроїв силової інфраструктури ЦОД
        end;

      //--------TCP структура---------------------

        PTMibTCPRow = ^TMibTCPRow;
        TMibTCPRow = packed record
          dwState: DWORD;
          dwLocalAddr: DWORD;
          dwLocalPort: DWORD;
          dwRemoteAddr: DWORD;
          dwRemotePort: DWORD;
        end;
        //
        PTMibTCPTable = ^TMibTCPTable;
        TMibTCPTable = packed record
          dwNumEntries: DWORD;
          Table: array[0..0] of TMibTCPRow;
        end;
        //
        PTMibTCPStats = ^TMibTCPStats;
        TMibTCPStats = packed record
          dwRTOAlgorithm: DWORD;
          dwRTOMin: DWORD;
          dwRTOMax: DWORD;
          dwMaxConn: DWORD;
          dwActiveOpens: DWORD;
```

```
    dwPassiveOpens: DWORD;
    dwAttemptFails: DWORD;
    dwEstabResets: DWORD;
    dwCurrEstab: DWORD;
    dwInSegs: DWORD;
    dwOutSegs: DWORD;
    dwRetransSegs: DWORD;
    dwInErrs: DWORD;
    dwOutRsts: DWORD;
    dwNumConns: DWORD;
  end;

//-----UDP СТРУКТУРА -------------------------

  PTMibUDPRow = ^TMibUDPRow;
  TMibUDPRow = packed record
    dwLocalAddr: DWORD;
    dwLocalPort: DWORD;
  end;
 //
  PTMibUDPTable = ^TMIBUDPTable;
  TMIBUDPTable = packed record
    dwNumEntries: DWORD;
    UDPTable: array[0..ANY_SIZE- 1] of TMibUDPRow;
  end;
 //
  PTMibUdpStats = ^TMIBUdpStats;
  TMIBUdpStats = packed record
    dwInDatagrams: DWORD;
    dwNoPorts: DWORD;
    dwInErrors: DWORD;
    dwOutDatagrams: DWORD;
    dwNumAddrs: DWORD;
  end;

//------IP СТРУКТУРА ------------------------

 //
  PTMibIPNetRow = ^TMibIPNetRow;
  TMibIPNetRow = packed record
    dwIndex: DWord;
    dwPhysAddrLen: DWord;
    bPhysAddr: TMACAddress;
    dwAddr: DWord;
    dwType: DWord;
  end;
  //
  PTMibIPNetTable = ^TMibIPNetTable;
  TMibIPNetTable = packed record
    dwNumEntries: DWORD;
    Table: array[0..ANY_SIZE- 1] of TMibIPNetRow;
  end;
  //
  PTMibIPStats = ^TMibIPStats;
  TMibIPStats = packed record
    dwForwarding: DWORD;
    dwDefaultTTL: DWORD;
    dwInReceives: DWORD;
    dwInHdrErrors: DWORD;
    dwInAddrErrors: DWORD;
    dwForwDatagrams: DWORD;
    dwInUnknownProtos: DWORD;
    dwInDiscards: DWORD;
    dwInDelivers: DWORD;
    dwOutRequests: DWORD;
    dwRoutingDiscards: DWORD;
    dwOutDiscards: DWORD;
    dwOutNoRoutes: DWORD;
    dwReasmTimeOut: DWORD;
```

```
    dwReasmReqds: DWORD;
    dwReasmOKs: DWORD;
    dwReasmFails: DWORD;
    dwFragOKs: DWORD;
    dwFragFails: DWORD;
    dwFragCreates: DWORD;
    dwNumIf: DWORD;
    dwNumAddr: DWORD;
    dwNumRoutes: DWORD;
  end;
  //
  PTMibIPAddrRow = ^TMibIPAddrRow;
  TMibIPAddrRow = packed record
    dwAddr: DWORD;
    dwIndex: DWORD;
    dwMask: DWORD;
    dwBCastAddr: DWORD;
    dwReasmSize: DWORD;
    Unused1,
    Unused2: WORD;
  end;
  //
  PTMibIPAddrTable = ^TMibIPAddrTable;
  TMibIPAddrTable = packed record
    dwNumEntries: DWORD;
    Table: array[0..ANY_SIZE- 1] of TMibIPAddrRow;
  end;

  //
  PTMibIPForwardRow = ^TMibIPForwardRow;
  TMibIPForwardRow = packed record
    dwForwardDest: DWORD;
    dwForwardMask: DWORD;
    dwForwardPolicy: DWORD;
    dwForwardNextHop: DWORD;
    dwForwardIFIndex: DWORD;
    dwForwardType: DWORD;
    dwForwardProto: DWORD;
    dwForwardAge: DWORD;
    dwForwardNextHopAS: DWORD;
    dwForwardMetric1: DWORD;
    dwForwardMetric2: DWORD;
    dwForwardMetric3: DWORD;
    dwForwardMetric4: DWORD;
    dwForwardMetric5: DWORD;
  end;
  //
  PTMibIPForwardTable = ^TMibIPForwardTable;
  TMibIPForwardTable = packed record
    dwNumEntries: DWORD;
    Table: array[0..ANY_SIZE- 1] of TMibIPForwardRow;
  end;

//---- ICMP-СТРУКТУРА ------------------------

  PTMibICMPStats = ^TMibICMPStats;
  TMibICMPStats = packed record
    dwMsgs: DWORD;
    dwErrors: DWORD;
    dwDestUnreachs: DWORD;
    dwTimeEcxcds: DWORD;
    dwParmProbs: DWORD;
    dwSrcQuenchs: DWORD;
    dwRedirects: DWORD;
    dwEchos: DWORD;
    dwEchoReps: DWORD;
    dwTimeStamps: DWORD;
    dwTimeStampReps: DWORD;
    dwAddrMasks: DWORD;
```

```
    dwAddrReps: DWORD;
  end;

  PTMibICMPInfo = ^TMibICMPInfo;
  TMibICMPInfo = packed record
    InStats: TMibICMPStats;
    OutStats: TMibICMPStats;
  end;

//---------імпорт до IPHLPAPI.DLL------------------

var

GetAdaptersInfo: function ( pAdapterInfo: PTIP_ADAPTER_INFO;
  pOutBufLen: PULONG ): DWORD; stdcall;

GetNetworkParams: function ( FixedInfo: PTFixedInfo; pOutPutLen: PULONG ):
        DWORD; stdcall;

GetTcpTable: function ( pTCPTable: PTMibTCPTable; pDWSize: PDWORD;
  bOrder: BOOL ): DWORD; stdcall;

GetTcpStatistics: function ( pStats: PTMibTCPStats ): DWORD; stdcall;

GetUdpTable: function ( pUdpTable: PTMibUDPTable; pDWSize: PDWORD;
 bOrder: BOOL ): DWORD; stdcall;

GetUdpStatistics: function ( pStats: PTMibUdpStats ): DWORD; stdcall;

GetIpStatistics: function ( pStats: PTMibIPStats ): DWORD; stdcall;

GetIpNetTable: function ( pIpNetTable: PTMibIPNetTable;
  pdwSize: PULONG;  bOrder: BOOL ): DWORD; stdcall;

GetIpAddrTable: function ( pIpAddrTable: PTMibIPAddrTable;
  pdwSize: PULONG; bOrder: BOOL ): DWORD; stdcall;

GetIpForwardTable: function ( pIPForwardTable: PTMibIPForwardTable;
  pdwSize: PULONG; bOrder: BOOL ): DWORD; stdCall;

GetIcmpStatistics: function ( pStats: PTMibICMPInfo ): DWORD; stdCall;

GetRTTAndHopCount: function ( DestIPAddress: DWORD; HopCount: PULONG;
  MaxHops: ULONG; RTT: PULONG ): BOOL; stdCall;

GetIfTable: function ( pIfTable: PTMibIfTable; pdwSize: PULONG;
  bOrder: boolean ): DWORD; stdCall;

GetIfEntry: function ( pIfRow: PTMibIfRow ): DWORD; stdCall;

// попередження – недокументована функція, можливі помилки при
використанні
  GetFriendlyIfIndex: function (var IfIndex: DWORD): DWORD; stdcall;


const
    IpHlpDLL = ' IPHLPAPI.DLL' ;
var
    IpHlpModule: THandle;

    function LoadIpHlp: Boolean;

implementation

function LoadIpHlp: Boolean;
begin
    Result := True;
    if IpHlpModule <> 0 then Exit;
```

```
      // відкрити DLL
          IpHlpModule := LoadLibrary (IpHlpDLL);
          if IpHlpModule = 0 then
          begin
              Result := false;
              exit ;
          end ;
          GetAdaptersInfo := GetProcAddress (IpHlpModule, ' GetAdaptersInfo' ) ;
          GetNetworkParams := GetProcAddress (IpHlpModule, ' GetNetworkParams' )
;
          GetTcpTable := GetProcAddress (IpHlpModule, ' GetTcpTable' ) ;
          GetTcpStatistics := GetProcAddress (IpHlpModule, ' GetTcpStatistics' )
;
          GetUdpTable := GetProcAddress (IpHlpModule, ' GetUdpTable' ) ;
          GetUdpStatistics := GetProcAddress (IpHlpModule, ' GetUdpStatistics' )
;
          GetIpStatistics := GetProcAddress (IpHlpModule, ' GetIpStatistics' ) ;
          GetIpNetTable := GetProcAddress (IpHlpModule, ' GetIpNetTable' ) ;
          GetIpAddrTable := GetProcAddress (IpHlpModule, ' GetIpAddrTable' ) ;
          GetIpForwardTable := GetProcAddress (IpHlpModule, ' GetIpForwardTable'
) ;
          GetIcmpStatistics := GetProcAddress (IpHlpModule, ' GetIcmpStatistics'
) ;
          GetRTTAndHopCount := GetProcAddress (IpHlpModule, ' GetRTTAndHopCount'
) ;
          GetIfTable := GetProcAddress (IpHlpModule, ' GetIfTable' ) ;
          GetIfEntry := GetProcAddress (IpHlpModule, ' GetIfEntry' ) ;
          GetFriendlyIfIndex := GetProcAddress (IpHlpModule, '
GetFriendlyIfIndex' ) ;
      end;

      initialization
          IpHlpModule := 0 ;
      finalization
          if IpHlpModule <> 0 then
          begin
              FreeLibrary (IpHlpModule) ;
              IpHlpModule := 0 ;
          end ;

      end.
```

Файл IPHelper.pas- функції роботи з IP-протоколом

```
unit IPHelper;


interface

uses
  Windows, Messages, SysUtils, Classes, Dialogs, IpHlpApi;

const
  NULL_IP       = '   0.0.  0.0' ;

//---перетворення добре відомих номерів портів до імен сервісів--------

type
  TWellKnownPort = record
    Prt: DWORD;
    Srv: string[20];
  end;


const
    // тільки найбільш популярні сервіси...
  WellKnownPorts: array[1..32] of TWellKnownPort
  = (
//    ( Prt: 0; Srv:   ' RESRVED'  ),     {Зарезервовано}
    ( Prt: 7; Srv:   ' ECHO   '  ),     {Ping    }
    ( Prt: 9; Srv:   ' DISCARD'  ),
    ( Prt: 13; Srv:  ' DAYTIME'  ),
    ( Prt: 17; Srv:  ' QOTD   '  ),     {Показчик на день}
    ( Prt: 19; Srv:  ' CHARGEN'  ),     {Генератор символів}
    ( Prt: 20; Srv:  ' FTPDATA'  ),     { File Transfer Protocol- дані}
    ( Prt: 21; Srv:  ' FTPCTRL'  ),     { File Transfer Protocol- управління}
    ( Prt: 22; Srv:  ' SSH    '  ),
    ( Prt: 23; Srv:  ' TELNET '  ),
    ( Prt: 25; Srv:  ' SMTP   '  ),     { Simple Mail Transfer Protocol}
    ( Prt: 37; Srv:  ' TIME   '  ),     { Часовий протокол }
    ( Prt: 43; Srv:  ' WHOIS  '  ),     { Сервіс – Кто це  }
    ( Prt: 53; Srv:  ' DNS    '  ),     { Domain Name Service }
    ( Prt: 67; Srv:  ' BOOTPS '  ),     { BOOTP Сервер }
    ( Prt: 68; Srv:  ' BOOTPC '  ),     { BOOTP Кієнт }
    ( Prt: 69; Srv:  ' TFTP   '  ),     { стандартний  FTP  }
    ( Prt: 70; Srv:  ' GOPHER '  ),     { Протокол Gopher       }
    ( Prt: 79; Srv:  ' FINGER '  ),     { Протокол Finger       }
    ( Prt: 80; Srv:  ' HTTP   '  ),     { Протокол HTTP         }
    ( Prt: 88; Srv:  ' KERBROS'  ),     { Протокол Kerberos     }
    ( Prt: 109; Srv: ' POP2   '  ),     { Протокол Post Office Protocol Version
2 }
    ( Prt: 110; Srv: ' POP3   '  ),     { Протокол Post Office Protocol Version
3 }
    ( Prt: 111; Srv: ' SUN_RPC'  ),     { Протокол SUN Remote Procedure Call }
    ( Prt: 119; Srv: ' NNTP   '  ),     { Протокол Network News Transfer
Protocol }
    ( Prt: 123; Srv: ' NTP    '  ),     { Протокол Network Time protocol
}
    ( Prt: 135; Srv: ' DCOMRPC'  ),     { Протокол Location Service
}
    ( Prt: 137; Srv: ' NBNAME '  ),     { NETBIOS сервіс імен          }
    ( Prt: 138; Srv: ' NBDGRAM'  ),     { NETBIOS сервіс датаграм     }
    ( Prt: 139; Srv: ' NBSESS '  ),     { NETBIOS сервіс сесій        }
    ( Prt: 143; Srv: ' IMAP   '  ),     { Протокол Internet Message Access
Protocol }
    ( Prt: 161; Srv: ' SNMP   '  ),     { Протокол Simple Netw. Management
Protocol }
    ( Prt: 169; Srv: ' SEND   '  )
```

```
    );


//------перетворення ICMP кодів помилок до рядків-------------

const
  ICMP_ERROR_BASE = 11000;
  IcmpErr : array[1..22] of string =
  (
   ' IP_BUFFER_TOO_SMALL' ,' IP_DEST_NET_UNREACHABLE' , '
IP_DEST_HOST_UNREACHABLE' ,
   ' IP_PROTOCOL_UNREACHABLE' , ' IP_DEST_PORT_UNREACHABLE' , ' IP_NO_RESOURCES'
,
   ' IP_BAD_OPTION' ,' IP_HARDWARE_ПОМИЛКА' , ' IP_PACKET_TOO_BIG' , '
IP_REQUEST_TIMED_OUT' ,
   ' IP_BAD_REQUEST' ,' IP_BAD_ROUTE' , ' IP_TTL_EXPIRED_TRANSIT' ,
   ' IP_TTL_EXPIRED_REASSEM' ,' IP_PARAMETER_PROBLEM' , ' IP_SOURCE_QUENCH' ,
   ' IP_OPTION_TOO_BIG' , ' IP_BAD_DESTINATION' ,' IP_ADDRESS_DELETED' ,
   ' IP_SPEC_MTU_CHANGE' , ' IP_MTU_CHANGE' , ' IP_UNLOAD'
  );


//-----Перетворення різних перерахованих величин у рядки--------

  ARPEntryType  : array[1..4] of string = ( ' інший' , ' неправильний' ,
    ' динамічний' , ' статичний'
    );
  TCPConnState  :
    array[1..12] of string =
    ( ' closed' , ' listening' , ' syn_sent' ,
    ' syn_rcvd' , ' established' , ' fin_wait1' ,
    ' fin_wait2' , ' close_wait' , ' closing' ,
    ' last_ack' , ' time_wait' , ' delete_tcb'
    );

  TCPToAlgo     : array[1..4] of string =
    ( ' Const.Timeout' , '  MIL-STD-1778' ,
    ' Van Jacobson' , ' інший' );

  IPForwTypes   : array[1..4] of string =
    ( ' інший' , ' invalid' , ' local' , ' remote'  );

  IPForwProtos  : array[1..18] of string =
    ( ' інший' , ' LOCAL' , ' NETMGMT' , ' ICMP' , ' EGP' ,
    ' GGP' , ' HELO' , ' RIP' , ' IS_IS' , ' ES_IS' ,
    ' CISCO' , ' BBN' , ' OSPF' , ' BGP' , ' BOOTP' ,
    ' AUTO_STAT' , ' STATIC' , ' NOT_DOD'  );

type

// для IpHlpNetworkParams
  TNetworkParams = record
    HostName: string ;
    DomainName: string ;
    CurrentDnsServer: string ;
    DnsServerTot: integer ;
    DnsServerNames: array [0..9] of string ;
    NodeType: UINT;
    ScopeID: string ;
    EnableRouting: UINT;
    EnableProxy: UINT;
    EnableDNS: UINT;
  end;

  TIfRows = array of TMibIfRow ; // динамічний масив колонок

// для IpHlpAdaptersInfo
  TAdaptorInfo = record
    AdapterName: string ;
```

```
    Description: string ;
    MacAddress: string ;
    Index: DWORD;
    aType: UINT;
    DHCPEnabled: UINT;
    CurrIPAddress: string ;
    CurrIPMask: string ;
    IPAddressTot: integer ;
    IPAddressList: array of string ;
    IPMaskList: array of string ;
    GatewayTot: integer ;
    GatewayList: array of string ;
    DHCPTot: integer ;
    DHCPServer: array of string ;
    HaveWINS: BOOL;
    PrimWINSTot: integer ;
    PrimWINSServer: array of string ;
    SecWINSTot: integer ;
    SecWINSServer: array of string ;
    LeaseObtained: LongInt ; // UNIX час, секунди з 1970
    LeaseExpires: LongInt;   // UNIX час, секунди з 1970
  end ;

  TAdaptorRows = array of TAdaptorInfo ;


//--------експортуємі дані---------------------

function IpHlpAdaptersInfo(var AdpTot: integer;var AdpRows: TAdaptorRows):
integer ;
procedure Get_AdaptersInfo( List: TStrings );
function IpHlpNetworkParams (var NetworkParams: TNetworkParams): integer ;
procedure Get_NetworkParams( List: TStrings );
procedure Get_ARPTable( List: TStrings );
procedure Get_TCPTable( List: TStrings );
procedure Get_TCPStatistics( List: TStrings );
function IpHlpTCPStatistics (var TCPStats: TMibTCPStats): integer ;
procedure Get_UDPTable( List: TStrings );
procedure Get_UDPStatistics( List: TStrings );
function IpHlpUdpStatistics (UdpStats: TMibUDPStats): integer ;
procedure Get_IPAddrTable( List: TStrings );
procedure Get_IPForwardTable( List: TStrings );
procedure Get_IPStatistics( List: TStrings );
function IpHlpIPStatistics (var IPStats: TMibIPStats): integer ;
function Get_RTTAndHopCount( IPAddr: DWORD; MaxHops: Longint;
  var RTT: longint; var HopCount: longint ): integer;
procedure Get_ICMPStats( ICMPIn, ICMPOut: TStrings );
function IpHlpIfTable(var IfTot: integer; var IfRows: TIfRows): integer ;
procedure Get_IfTable( List: TStrings );
function IpHlpIfEntry(Index: integer; var IfRow: TMibIfRow): integer ;
procedure Get_RecentDestIPs( List: TStrings );

// функції перетворення
function MacAddr2Str( MacAddr: TMacAddress; size: integer ): string;
function IpAddr2Str( IPAddr: DWORD ): string;
function Str2IpAddr( IPStr: string ): DWORD;
function Port2Str( nwoPort: DWORD ): string;
function Port2Wrd( nwoPort: DWORD ): DWORD;
function Port2Svc( Port: DWORD ): string;
function ICMPErr2Str( ICMPErrCode: DWORD) : string;

implementation

var
  RecentIPs     : TStringList;

//-------Основні функції----------------------

{ отримання наступного "токена" з рядка }
```

```
function NextToken( var s: string; Separator: char ): string;
var
  Sep_Pos       : byte;
begin
  Result := ' ' ;
  if length( s ) > 0 then begin
    Sep_Pos := pos( Separator, s );
    if Sep_Pos > 0 then begin
      Result := copy( s, 1, Pred( Sep_Pos ) );
      Delete( s, 1, Sep_Pos );
    end
    else begin
      Result := s;
      s := ' ' ;
    end;
  end;
end;

//-----------------------------------
{ перетворення числового MAC-адреса  до ww-xx-yy-zz рядка }
function MacAddr2Str( MacAddr: TMacAddress; size: integer ): string;
var
  i             : integer;
begin
  if Size = 0 then
  begin
    Result := '  00-00-00' ;
    EXIT;
  end
  else Result := ' ' ;
  //
  for i := 1 to Size do
    Result := Result + IntToHex( MacAddr[i], 2) + ' -' ;
  Delete( Result, Length( Result ), 1 );
end;

//-----------------------------------
{ перетворення IP-адреси в мережний байт типу DWORD }
function IpAddr2Str( IPAddr: DWORD ): string;
var
  i             : integer;
begin
  Result := ' ' ;
  for i := 1 to 4 do
  begin
    Result := Result + Format( ' %3d.' , [IPAddr and $FF] );
    IPAddr := IPAddr shr 8;
  end;
  Delete( Result, Length( Result ), 1 );
end;

//-----------------------------------
{ перетворення крапкової десяткової IP-адреси в мережний байт типу DWORD}
function Str2IpAddr( IPStr: string ): DWORD;
var
  i             : integer;
  Num           : DWORD;
begin
  Result := 0;
  for i := 1 to 4 do
  try
    Num := ( StrToInt( NextToken( IPStr, ' .'  ) ) ) shl 24;
    Result := ( Result shr 8 ) or Num;
  except
    Result := 0;
  end;

end;
```

```
//-----------------------------------
{ перетворення номеру порту в мережний байт типу DWORD }
function Port2Wrd( nwoPort: DWORD ): DWORD;
begin
  Result := Swap( WORD( nwoPort ) );
end;

//-----------------------------------
{ перетворення номеру порту в мережний байт типу string }
function Port2Str( nwoPort: DWORD ): string;
begin
  Result := IntToStr( Port2Wrd( nwoPort ) );
end;

//-----------------------------------
{ перетворення номеру порту в сервіс ID }
function Port2Svc( Port: DWORD ): string;
var
  i             : integer;
begin
  Result := Format( ' %4d' , [Port] ); // у випадку, якщо порт не знайдено
  for i := Low( WellKnownPorts ) to High( WellKnownPorts ) do
    if Port = WellKnownPorts[i].Prt then
    begin
      Result := WellKnownPorts[i].Srv;
      BREAK;
    end;
end;

//-----------------------------------
{ голона частина, фіксація мережний параметрів мережі IoT пристроїв силової
інфраструктури ЦОД}

procedure Get_NetworkParams( List: TStrings );
var
    NetworkParams: TNetworkParams ;
    I, ErrorCode: integer ;
begin
    if not Assigned( List ) then EXIT;
    List.Clear;
    ErrorCode := IpHlpNetworkParams (NetworkParams) ;
    if ErrorCode <> 0 then
    begin
        List.Add (SysErrorMessage (ErrorCode));
        exit;
    end ;
    with NetworkParams do
    begin
        List.Add( ' Ім'я хосту          : '  + HostName );
        List.Add( ' Домен            : '  + DomainName );
        List.Add( ' NETBIOS тип : '  + NETBIOSTypes[NodeType] );
        List.Add( ' DHCP область        : '  + ScopeID );
        List.Add( ' ROUTING визначено   : '  + IntToStr( EnableRouting ) );
        List.Add( ' PROXY   визначено   : '  + IntToStr( EnableProxy ) );
        List.Add( ' DNS     визначено   : '  + IntToStr( EnableDNS ) );
        if DnsServerTot <> 0 then
        begin
            for I := 0 to Pred (DnsServerTot) do
                List.Add( ' DNS адреса серверу   : '  + DnsServerNames [I] ) ;
        end ;
    end ;
end ;

//-----------------------------------//
function IpHlpNetworkParams (var NetworkParams: TNetworkParams): integer ;
var
  FixedInfo     : PTFixedInfo;         // дані
  InfoSize      : Longint;
  PDnsServer    : PTIP_ADDR_STRING ;   // дані
```

```
begin
    InfoSize := 0 ;   // дані
    result := ERROR_NOT_SUPPORTED ;
    if NOT LoadIpHlp then exit ;
    result := GetNetworkParams( Nil, @InfoSize );  // дані
    if result <> ERROR_BUFFER_OVERFLOW then exit ; // дані
    GetMem (FixedInfo, InfoSize) ;                    // дані
    try
    result := GetNetworkParams( FixedInfo, @InfoSize );   // дані
    if result <> ERROR_SUCCESS then exit ;
    NetworkParams.DnsServerTot := 0 ;
    with FixedInfo^ do
    begin
        NetworkParams.HostName := trim (HostName) ;
        NetworkParams.DomainName := trim (DomainName) ;
        NetworkParams.ScopeId := trim (ScopeID) ;
        NetworkParams.NodeType := NodeType ;
        NetworkParams.EnableRouting := EnableRouting ;
        NetworkParams.EnableProxy := EnableProxy ;
        NetworkParams.EnableDNS := EnableDNS ;
        NetworkParams.DnsServerNames [0] := DNSServerList.IPAddress ;  // дані
        if NetworkParams.DnsServerNames [0] <> ' '  then
                                    NetworkParams.DnsServerTot := 1 ;
        PDnsServer := DnsServerList.Next;
        while PDnsServer <> Nil do
        begin
            NetworkParams.DnsServerNames [NetworkParams.DnsServerTot] :=
                                    PDnsServer^.IPAddress ;  // дані
            inc (NetworkParams.DnsServerTot) ;
            if NetworkParams.DnsServerTot >=
                          Length (NetworkParams.DnsServerNames) then exit ;
            PDnsServer := PDnsServer.Next ;
        end;
    end ;
    finally
       FreeMem (FixedInfo) ;                    // дані
    end ;
end;

//-----------------------------------

function ICMPErr2Str( ICMPErrCode: DWORD) : string;
begin
   Result := ' UnknownError : '  + IntToStr( ICMPErrCode );
   dec( ICMPErrCode, ICMP_ERROR_BASE );
   if ICMPErrCode in [Low(ICMpErr)..High(ICMPErr)] then
     Result := ICMPErr[ ICMPErrCode];
end;


//-----------------------------------

// включення байтів у/з для кожного адаптера

function IpHlpIfTable(var IfTot: integer; var IfRows: TIfRows): integer ;
var
  I,
  TableSize   : integer;
  pBuf, pNext : PChar;
begin
  result := ERROR_NOT_SUPPORTED ;
  if NOT LoadIpHlp then exit ;
  SetLength (IfRows, 0) ;
  IfTot := 0 ; // дані
  TableSize := 0;
   // перший виклик: необхідно отримати розмір пам' яті
  result := GetIfTable (Nil, @TableSize, false) ;  // дані
  if result <> ERROR_INSUFFICIENT_BUFFER then exit ;
  GetMem( pBuf, TableSize );
```

```
  try
      FillChar (pBuf^, TableSize, #0);  // очищаємо буфер з  W98 не беремо
крапку таблиці
      result := GetIfTable (PTMibIfTable (pBuf), @TableSize, false) ;
      if result <> NO_ERROR then exit ;
      IfTot := PTMibIfTable (pBuf)^.dwNumEntries ;
      if IfTot = 0 then exit ;
      SetLength (IfRows, IfTot) ;
      pNext := pBuf + SizeOf(IfTot) ;
      for i := 0 to Pred (IfTot) do
      begin
         IfRows [i] := PTMibIfRow (pNext )^ ;
         inc (pNext, SizeOf (TMibIfRow)) ;
      end;
  finally
      FreeMem (pBuf) ;
  end ;
end;

procedure Get_IfTable( List: TStrings );
var
  IfRows        : TIfRows ;
  Error, I      : integer;
  NumEntries    : integer;
  sDescr, sIfName: string ;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  SetLength (IfRows, 0) ;
  Error := IpHlpIfTable (NumEntries, IfRows) ;
  if (Error <> 0) then
      List.Add( SysErrorMessage( GetLastError ) )
  else if NumEntries = 0 then
      List.Add( ' данних немає ' )
  else
  begin
      for I := 0 to Pred (NumEntries) do
      begin
          with IfRows [I] do
          begin
             if wszName [1] = #0 then
                 sIfName := ' '
             else
                 sIfName := WideCharToString (@wszName) ;  // конвертуємо Юнікод
до рядка
             sIfName := trim (sIfName) ;
             sDescr := bDescr ;
             sDescr := trim (sDescr);
             List.Add (Format (
              ' %0.8x |%3d | %16s |%8d |%12d |%2d |%2d |%10d |%10d | %-s| %-s'
,
               [dwIndex, dwType, MacAddr2Str( TMacAddress( bPhysAddr ),
               dwPhysAddrLen ), dwMTU, dwSpeed, dwAdminStatus,
               dwOPerStatus, Int64 (dwInOctets), Int64 (dwOutOctets),  //
конвертуємо до 32-біт
               sIfName, sDescr] )  // дані, додані в/з
               );
          end;
      end ;
  end ;
  SetLength (IfRows, 0) ;  // вільна пам' ять
end ;

function IpHlpIfEntry(Index: integer; var IfRow: TMibIfRow): integer ;
begin
  result := ERROR_NOT_SUPPORTED ;
  if NOT LoadIpHlp then exit ;
  FillChar (IfRow, SizeOf (TMibIfRow), #0);  // очищаємо буфер з  W98 не беремо
  IfRow.dwIndex := Index ;
```

```
  result := GetIfEntry (@IfRow) ;
end ;

//-----------------------------------
{ інформація про інстальовані адаптери }

function IpHlpAdaptersInfo(var AdpTot: integer; var AdpRows: TAdaptorRows):
integer ;
var
  BufLen        : DWORD;
  AdapterInfo   : PTIP_ADAPTER_INFO;
  PIpAddr       : PTIP_ADDR_STRING;
  PBuf          : PCHAR ;
  I             : integer ;
begin
  SetLength (AdpRows, 4) ;
  AdpTot := 0 ;
  BufLen := 0 ;
  result := GetAdaptersInfo( Nil, @BufLen );
  if (result <> ERROR_INSUFFICIENT_BUFFER) and (result = NO_ERROR) then exit ;
  GetMem( pBuf, BufLen );
  try
      FillChar (pBuf^, BufLen, #0);  // очищуємо буфер
      result := GetAdaptersInfo( PTIP_ADAPTER_INFO (PBuf), @BufLen );
      if result = NO_ERROR then
      begin
         AdapterInfo := PTIP_ADAPTER_INFO (PBuf) ;
         while ( AdapterInfo <> nil ) do
         begin
            AdpRows [AdpTot].IPAddressTot := 0 ;
            SetLength (AdpRows [AdpTot].IPAddressList, 2) ;
            SetLength (AdpRows [AdpTot].IPMaskList, 2) ;
            AdpRows [AdpTot].GatewayTot := 0 ;
            SetLength (AdpRows [AdpTot].GatewayList, 2) ;
            AdpRows [AdpTot].DHCPTot := 0 ;
            SetLength (AdpRows [AdpTot].DHCPServer, 2) ;
            AdpRows [AdpTot].PrimWINSTot := 0 ;
            SetLength (AdpRows [AdpTot].PrimWINSServer, 2) ;
            AdpRows [AdpTot].SecWINSTot := 0 ;
            SetLength (AdpRows [AdpTot].SecWINSServer, 2) ;
            AdpRows [AdpTot].CurrIPAddress := NULL_IP;
            AdpRows [AdpTot].CurrIPMask := NULL_IP;
            AdpRows [AdpTot].AdapterName := Trim( string(
AdapterInfo^.AdapterName ) );
            AdpRows [AdpTot].Description := Trim( string(
AdapterInfo^.Description ) );
            AdpRows [AdpTot].MacAddress := MacAddr2Str( TMacAddress(
                                AdapterInfo^.Address ),
AdapterInfo^.AddressLength ) ;
            AdpRows [AdpTot].Index := AdapterInfo^.Index ;
            AdpRows [AdpTot].aType := AdapterInfo^.aType ;
            AdpRows [AdpTot].DHCPEnabled := AdapterInfo^.DHCPEnabled ;
            if AdapterInfo^.CurrentIPAddress <> Nil then
            begin
                AdpRows [AdpTot].CurrIPAddress :=
AdapterInfo^.CurrentIPAddress.IpAddress ;
                AdpRows [AdpTot].CurrIPMask :=
AdapterInfo^.CurrentIPAddress.IpMask ;
            end ;

        // беремо список IP адрес та конвертуємо в IPAddressList
            I := 0 ;
            PIpAddr := @AdapterInfo^.IPAddressList ;
            while (PIpAddr <> Nil) do
            begin
                AdpRows [AdpTot].IPAddressList [I] := PIpAddr.IpAddress ;
                AdpRows [AdpTot].IPMaskList [I] := PIpAddr.IpMask ;
                PIpAddr := PIpAddr.Next ;
                inc (I) ;
```

```
            if Length (AdpRows [AdpTot].IPAddressList) <= I then
            begin
                 SetLength (AdpRows [AdpTot].IPAddressList, I -2) ;
                 SetLength (AdpRows [AdpTot].IPMaskList, I -2) ;
            end ;
        end ;
        AdpRows [AdpTot].IPAddressTot := I ;

    // беремо список IP адрес для GatewayList
        I := 0 ;
        PIpAddr := @AdapterInfo^.GatewayList ;
        while (PIpAddr <> Nil) do
        begin
            AdpRows [AdpTot].GatewayList [I] := PIpAddr.IpAddress ;
            PIpAddr := PIpAddr.Next ;
            inc (I) ;
            if Length (AdpRows [AdpTot].GatewayList) <= I then
                        SetLength (AdpRows [AdpTot].GatewayList, I -2) ;
        end ;
        AdpRows [AdpTot].GatewayTot := I ;

    // беремо список IP адрес для  GatewayList
        I := 0 ;
        PIpAddr := @AdapterInfo^.DHCPServer ;
        while (PIpAddr <> Nil) do
        begin
            AdpRows [AdpTot].DHCPServer [I] := PIpAddr.IpAddress ;
            PIpAddr := PIpAddr.Next ;
            inc (I) ;
            if Length (AdpRows [AdpTot].DHCPServer) <= I then
                        SetLength (AdpRows [AdpTot].DHCPServer, I -2) ;
        end ;
        AdpRows [AdpTot].DHCPTot := I ;

    // беремо список IP адрес для PrimaryWINSServer
        I := 0 ;
        PIpAddr := @AdapterInfo^.PrimaryWINSServer ;
        while (PIpAddr <> Nil) do
        begin
            AdpRows [AdpTot].PrimWINSServer [I] := PIpAddr.IpAddress ;
            PIpAddr := PIpAddr.Next ;
            inc (I) ;
            if Length (AdpRows [AdpTot].PrimWINSServer) <= I then
                        SetLength (AdpRows [AdpTot].PrimWINSServer, I -2) ;
        end ;
        AdpRows [AdpTot].PrimWINSTot := I ;

    // беремо список IP адрес для SecondaryWINSServer
        I := 0 ;
        PIpAddr := @AdapterInfo^.SecondaryWINSServer ;
        while (PIpAddr <> Nil) do
        begin
            AdpRows [AdpTot].SecWINSServer [I] := PIpAddr.IpAddress ;
            PIpAddr := PIpAddr.Next ;
            inc (I) ;
            if Length (AdpRows [AdpTot].SecWINSServer) <= I then
                        SetLength (AdpRows [AdpTot].SecWINSServer, I -2) ;
        end ;
        AdpRows [AdpTot].SecWINSTot := I ;

        AdpRows [AdpTot].LeaseObtained := AdapterInfo^.LeaseObtained ;
        AdpRows [AdpTot].LeaseExpires := AdapterInfo^.LeaseExpires ;

        inc (AdpTot) ;
        if Length (AdpRows) <= AdpTot then
                        SetLength (AdpRows, AdpTot -2) ;  // більше пам' яті
        AdapterInfo := AdapterInfo^.Next;
    end ;
    SetLength (AdpRows, AdpTot) ;
```

```
      end ;
  finally
      FreeMem( pBuf );
  end ;
end ;

procedure Get_AdaptersInfo( List: TStrings );
var
  AdpTot: integer;
  AdpRows: TAdaptorRows ;
  Error: DWORD ;
  I: integer ;
  //J: integer ;
  //S: string ;        id.
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  SetLength (AdpRows, 0) ;
  AdpTot := 0 ;
  Error := IpHlpAdaptersInfo(AdpTot, AdpRows) ;
  if (Error <> 0) then
      List.Add( SysErrorMessage( GetLastError ) )
  else if AdpTot = 0 then
      List.Add( ' данних немає ’  )
  else
  begin
      for I := 0 to Pred (AdpTot) do
      begin
        with AdpRows [I] do
        begin
            //List.Add(AdapterName + ' |’  + Description ); // jpt : не
використовується
            List.Add( Format(' %8.8x | %6s | %16s | %2d | %16s | %16s | %16s’ ,
                [Index, AdaptTypes[aType], MacAddress, DHCPEnabled,
                GatewayList [0], DHCPServer [0], PrimWINSServer [0]] ) );
            {if IPAddressTot <> 0 then    // jpt : не використовується
            begin
                S := ' '  ;
                for J := 0 to Pred (IPAddressTot) do
                        S := S + IPAddressList [J] + ' /’  + IPMaskList [J] + '
| '  ;
                List.Add(IntToStr (IPAddressTot) + '  IP Addresse(s): '  + S);
            end ;
            List.Add( '   ' ); }
        end ;
      end ;
  end ;
  SetLength (AdpRows, 0) ;
end ;

//-----------------------------------
{ моніторимо час доступу до IP мережі IoT пристроїв силової інфраструктури ЦОД}
function Get_RTTAndHopCount( IPAddr: DWORD; MaxHops: Longint; var RTT: Longint;
  var HopCount: Longint ): integer;
begin
  if not GetRTTAndHopCount( IPAddr, @HopCount, MaxHops, @RTT ) then
  begin
    Result := GetLastError;
    RTT :=-1; // Расположення BAD_HOST_NAME,etc...
    HopCount :=-1;
  end
  else
    Result := NO_ERROR;
end;

//-----------------------------------
{ ARP-таблиця включає відношення між віддаленим IP та віддаленим MAC-адресом.
}
procedure Get_ARPTable( List: TStrings );
```

```
var
  IPNetRow      : TMibIPNetRow;
  TableSize     : DWORD;
  NumEntries    : DWORD;
  ErrorCode     : DWORD;
  i             : integer;
  pBuf          : PChar;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  // перший виклик: беремо довжину таблиці
  TableSize := 0;
  ErrorCode := GetIPNetTable( Nil, @TableSize, false );   // дані
  //
  if ErrorCode = ERROR_NO_DATA then
  begin
    List.Add( '  ARP-кеш пустий.'  );
    EXIT;
  end;
  // беремо таблицю
  GetMem( pBuf, TableSize );
  NumEntries := 0 ;
  try
  ErrorCode := GetIpNetTable( PTMIBIPNetTable( pBuf ), @TableSize, false );
  if ErrorCode = NO_ERROR then
  begin
    NumEntries := PTMIBIPNetTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      inc( pBuf, SizeOf( DWORD ) ); // беремо розмір останьої таблиці
      for i := 1 to NumEntries do
      begin
        IPNetRow := PTMIBIPNetRow( PBuf )^;
        with IPNetRow do
          List.Add( Format( ' %8x | %12s | %16s | %10s' ,
                            [dwIndex, MacAddr2Str( bPhysAddr, dwPhysAddrLen ),
                            IPAddr2Str( dwAddr ), ARPEntryType[dwType]
                            ]));
        inc( pBuf, SizeOf( IPNetRow ) );
      end;
    end
    else
      List.Add( '  ARP-кеш пустий.'  );
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );

  // необхідно відновити показник!
  finally
      dec( pBuf, SizeOf( DWORD ) + NumEntries -SizeOf( IPNetRow ) );
      FreeMem( pBuf );
  end ;
end;


//-----------------------------------
procedure Get_TCPTable( List: TStrings );
var
  TCPRow        : TMIBTCPRow;
  i,
    NumEntries  : integer;
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  DestIP        : string;
  pBuf          : PChar;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  RecentIPs.Clear;
```

```
  // перший виклик: беремо довжину таблиці
  TableSize := 0;
  NumEntries := 0 ;
  ErrorCode := GetTCPTable(Nil, @TableSize, false );  // дані
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  // беремо розмір пам' яті, викликаємо знову
  GetMem( pBuf, TableSize );
  // беремо таблицю
  ErrorCode := GetTCPTable( PTMIBTCPTable( pBuf ), @TableSize, false );
  if ErrorCode = NO_ERROR then
  begin

    NumEntries := PTMIBTCPTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      inc( pBuf, SizeOf( DWORD ) ); // беремо розмір останьої таблиці
      for i := 1 to NumEntries do
      begin
        TCPRow := PTMIBTCPRow( pBuf )^; // беремо останій запис
        with TCPRow do
        begin
          if dwRemoteAddr = 0 then
            dwRemotePort := 0;
          DestIP := IPAddr2Str( dwRemoteAddr );
          List.Add(
            Format( ' %15s : %-7s | %15s : %-7s | %-16s' ,
            [IpAddr2Str( dwLocalAddr ),
            Port2Svc( Port2Wrd( dwLocalPort ) ),
              DestIP,
              Port2Svc( Port2Wrd( dwRemotePort ) ),
              TCPConnState[dwState]
              ] ) );
         //
            if (not ( dwRemoteAddr = 0 ))
            and ( RecentIps.IndexOf(DestIP) =-1 ) then
               RecentIPs.Add( DestIP );
        end;
        inc( pBuf, SizeOf( TMIBTCPRow ) );
      end;
    end;
  end
  else
    List.Add( SyserrorMessage( ErrorCode ) );
  dec( pBuf, SizeOf( DWORD ) + NumEntries -SizeOf( TMibTCPRow ) );
  FreeMem( pBuf );
end;

//---------------------------------
procedure Get_TCPStatistics( List: TStrings );
var
  TCPStats      : TMibTCPStats;
  ErrorCode     : DWORD;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  if NOT LoadIpHlp then exit ;
  ErrorCode := GetTCPStatistics( @TCPStats );
  if ErrorCode = NO_ERROR then
    with TCPStats do
    begin
      List.Add( ' Алгоритм повторної передачі : '  + TCPToAlgo[dwRTOAlgorithm]
);
      List.Add( ' Мінімальний час виходу         : '  + IntToStr( dwRTOMin ) + '
ms'  );
      List.Add( ' Максимальний час виходу         : '  + IntToStr( dwRTOMax ) +
'  ms'  );
```

```
      List.Add( ' Максимальне число підключень : '  + IntToStr( dwRTOAlgorithm )
);
      List.Add( ' Активні підключення             : '  + IntToStr( dwActiveOpens
) );
      List.Add( ' пасивні підключення            : '  + IntToStr( dwPassiveOpens
) );
      List.Add( ' Невдала спроба відкриття     : '  + IntToStr( dwAttemptFails )
);
      List.Add( ' Скидання встановленого підключення  : '  + IntToStr(
dwEstabResets ) );
      List.Add( ' Поточне встановлене підключення.: '  + IntToStr( dwCurrEstab )
);
      List.Add( ' Отримані сегменти        : '  + IntToStr( dwInSegs ) );
      List.Add( ' Передані сегменти            : '  + IntToStr( dwOutSegs ) );
      List.Add( ' Перепідключені сегменти   : '  + IntToStr( dwReTransSegs ) );
      List.Add( ' помилка входу          : '  + IntToStr( dwInErrs ) );
      List.Add( ' Перезавантаження вихідних          : '  + IntToStr( dwOutRsts
) );
      List.Add( ' Совокупні зв' язки    : '  + IntToStr( dwNumConns ) );
    end
  else
    List.Add( SyserrorMessage( ErrorCode ) );
end;

function IpHlpTCPStatistics (var TCPStats: TMibTCPStats): integer ;
begin
    result := ERROR_NOT_SUPPORTED ;
    if NOT LoadIpHlp then exit ;
    result := GetTCPStatistics( @TCPStats );
end;

//-----------------------------------
procedure Get_UDPTable( List: TStrings );
var
  UDPRow        : TMIBUDPRow;
  i,
    NumEntries  : integer;
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  pBuf          : PChar;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;

  // перший виклик: беремо довжину таблиці
  TableSize := 0;
  NumEntries := 0 ;
  ErrorCode := GetUDPTable(Nil, @TableSize, false );
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  // виділяємо пам' ять, викликаємо знову
  GetMem( pBuf, TableSize );

  // беремо таблицю
  ErrorCode := GetUDPTable( PTMIBUDPTable( pBuf ), @TableSize, false );
  if ErrorCode = NO_ERROR then
  begin
    NumEntries := PTMIBUDPTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      inc( pBuf, SizeOf( DWORD ) ); // беремо розмір останьої таблиці
      for i := 1 to NumEntries do
      begin
        UDPRow := PTMIBUDPRow( pBuf )^; // беремо останій запис
        with UDPRow do
          List.Add( Format( ' %15s : %-6s' ,
            [IpAddr2Str( dwLocalAddr ),
            Port2Svc( Port2Wrd( dwLocalPort ) )
```

```
              ] ) );
          inc( pBuf, SizeOf( TMIBUDPRow ) );
        end;
      end
      else
        List.Add( ' немає даних.'  );
    end
    else
      List.Add( SyserrorMessage( ErrorCode ) );
    dec( pBuf, SizeOf( DWORD ) + NumEntries -SizeOf( TMibUDPRow ) );
    FreeMem( pBuf );
end;

//-----------------------------------
procedure Get_IPAddrTable( List: TStrings );
var
  IPAddrRow     : TMibIPAddrRow;
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  i             : integer;
  pBuf          : PChar;
  NumEntries    : DWORD;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  TableSize := 0; ;
  NumEntries := 0 ;
  // перший виклик: беремо довжину таблиці
  ErrorCode := GetIpAddrTable(Nil, @TableSize, true );  // дані
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  GetMem( pBuf, TableSize );
  // беремо таблицю
  ErrorCode := GetIpAddrTable( PTMibIPAddrTable( pBuf ), @TableSize, true );
  if ErrorCode = NO_ERROR then
  begin
    NumEntries := PTMibIPAddrTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      inc( pBuf, SizeOf( DWORD ) );
      for i := 1 to NumEntries do
      begin
        IPAddrRow := PTMIBIPAddrRow( pBuf )^;
        with IPAddrRow do
          List.Add( Format( ' %8.8x | %15s | %15s | %15s | %8.8d' ,
            [dwIndex,
            IPAddr2Str( dwAddr ),
              IPAddr2Str( dwMask ),
              IPAddr2Str( dwBCastAddr ),
              dwReasmSize
              ] ) );
        inc( pBuf, SizeOf( TMIBIPAddrRow ) );
      end;
    end
    else
      List.Add( ' немає даних.'  );
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );

  // відновлюємо показчик!
  dec( pBuf, SizeOf( DWORD ) + NumEntries -SizeOf( IPAddrRow ) );
  FreeMem( pBuf );
end;

//-----------------------------------
{ отримуємо дані з таблиці маршрутизації мережі IoT пристроїв силової
інфраструктури ЦОД; }
```

```
procedure Get_IPForwardTable( List: TStrings );
var
  IPForwRow     : TMibIPForwardRow;
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  i             : integer;
  pBuf          : PChar;
  NumEntries    : DWORD;
begin

  if not Assigned( List ) then EXIT;
  List.Clear;
  TableSize := 0;

  // перший виклик: беремо довжину таблиці
  NumEntries := 0 ;
  ErrorCode := GetIpForwardTable(Nil, @TableSize, true);
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  // беремо таблицю
  GetMem( pBuf, TableSize );
  ErrorCode := GetIpForwardTable( PTMibIPForwardTable( pBuf ), @TableSize,
true);
  if ErrorCode = NO_ERROR then
  begin
    NumEntries := PTMibIPForwardTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      inc( pBuf, SizeOf( DWORD ) );
      for i := 1 to NumEntries do
      begin
        IPForwRow := PTMibIPForwardRow( pBuf )^;
        with IPForwRow do
        begin
          if (dwForwardType < 1)
          or (dwForwardType > 4) then
                   dwForwardType := 1 ;   // дані
          List.Add( Format(
            ' %15s | %15s | %15s | %8.8x | %7s |   %5.5d |  %7s |   %2.2d' ,
            [IPAddr2Str( dwForwardDest ),
            IPAddr2Str( dwForwardMask ),
              IPAddr2Str( dwForwardNextHop ),
              dwForwardIFIndex,
              IPForwTypes[dwForwardType],
              dwForwardNextHopAS,
              IPForwProtos[dwForwardProto],
              dwForwardMetric1
              ] ) );
        end ;
        inc( pBuf, SizeOf( TMibIPForwardRow ) );
      end;
    end
    else
      List.Add( ' немає даних.'  );
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );
  dec( pBuf, SizeOf( DWORD ) + NumEntries -SizeOf( TMibIPForwardRow ) );
  FreeMem( pBuf );
end;

//-----------------------------------
procedure Get_IPStatistics( List: TStrings );
var
  IPStats       : TMibIPStats;
  ErrorCode     : integer;
begin
  if not Assigned( List ) then EXIT;
```

```
  if NOT LoadIpHlp then exit ;
  ErrorCode := GetIPStatistics( @IPStats );
  if ErrorCode = NO_ERROR then
  begin
    List.Clear;
    with IPStats do
    begin
      if dwForwarding = 1 then
        List.add( ' Розблокована пересилка      : '  + ' так'  )
      else
        List.add(' Розблокована пересилка      : '  + ' ні'  );
      List.add( ' Любий TTL             : '  + inttostr( dwDefaultTTL ) );
      List.add( ' Датаграмма прийнята     : '  + inttostr( dwInReceives ) );
      List.add( ' Помилка заголовку      (In)  : '  + inttostr( dwInHdrErrors )
);
      List.add( ' Помилка адреси    (In)  : '  + inttostr( dwInAddrErrors ) );
      List.add( ' Датаграма переслана     : '  + inttostr( dwForwDatagrams ) );
// дані
      List.add( ' Невизначений протокол (In)  : '  + inttostr( dwInUnknownProtos
) );
      List.add( ' Датаграмма відмовлена     : '  + inttostr( dwInDiscards ) );
      List.add( ' Датаграма встановлена     : '  + inttostr( dwInDelivers ) );
      List.add( ' Зовнішній запрос            : '  + inttostr( dwOutRequests )
);
      List.add( ' Маршрутизація не виконана      : '  + inttostr(
dwRoutingDiscards ) );
      List.add( ' Немає маршрутів        (Out)  : '  + inttostr( dwOutNoRoutes )
);
      List.add( ' Перебраний час     : '  + inttostr( dwReasmTimeOut ) );
      List.add( ' Запит перебору     : '  + inttostr( dwReasmReqds ) );
      List.add( ' Повний перебор : '  + inttostr( dwReasmOKs ) );
      List.add( ' Помилка перебору     : '  + inttostr( dwReasmFails ) );
      List.add( ' Повна фрагментація: '  + inttostr( dwFragOKs ) );
      List.add( ' Помилка фрагментації   : '  + inttostr( dwFragFails ) );
      List.add( ' Датаграмма фрагментована    : '  + inttostr( dwFRagCreates )
);
      List.add( ' Кількість інтерфейсів    : '  + inttostr( dwNumIf ) );
      List.add( ' Кількість IP-адрес  : '  + inttostr( dwNumAddr ) );
      List.add( ' Маршрут в таблиці маршрутизатора  : '  + inttostr( dwNumRoutes
) );
    end;
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );
end;

function IpHlpIPStatistics (var IPStats: TMibIPStats): integer ;      // дані
begin
    result := ERROR_NOT_SUPPORTED ;
    if NOT LoadIpHlp then exit ;
    result := GetIPStatistics( @IPStats );
end ;

//-----------------------------------
procedure Get_UdpStatistics( List: TStrings );
var
  UdpStats      : TMibUDPStats;
  ErrorCode     : integer;
begin
  if not Assigned( List ) then EXIT;
  ErrorCode := GetUDPStatistics( @UdpStats );
  if ErrorCode = NO_ERROR then
  begin
    List.Clear;
    with UDPStats do
    begin
      List.add( ' Датаграми (In)     : '  + inttostr( dwInDatagrams ) );
      List.add( '  Датаграми (Out)    : '  + inttostr( dwOutDatagrams ) );
      List.add( ' Немає портів          : '  + inttostr( dwNoPorts ) );
```

```
      List.add( ' Помилка    (In)    : '  + inttostr( dwInErrors ) );
      List.add( ' UDP список портів  : '  + inttostr( dwNumAddrs ) );
    end;
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );
end;

//--------------------------------------*//
function IpHlpUdpStatistics (UdpStats: TMibUDPStats): integer ;     // дані
begin
    result := ERROR_NOT_SUPPORTED ;
    if NOT LoadIpHlp then exit ;
    result := GetUDPStatistics (@UdpStats) ;
end ;

//-----------------------------------
procedure Get_ICMPStats( ICMPIn, ICMPOut: TStrings );
var
  ErrorCode     : DWORD;
  ICMPStats     : PTMibICMPInfo;
begin
  if ( ICMPIn = nil ) or ( ICMPOut = nil ) then EXIT;
  ICMPIn.Clear;
  ICMPOut.Clear;
  New( ICMPStats );
  ErrorCode := GetICMPStatistics( ICMPStats );
  if ErrorCode = NO_ERROR then
  begin
    with ICMPStats.InStats do
    begin
      ICMPIn.Add( '  Прийнято повідомлень    : '  + IntToStr( dwMsgs ) );
      ICMPIn.Add( ' Помилка                  : '  + IntToStr( dwErrors ) );
      ICMPIn.Add( ' Розташування недосягенно    : '  + IntToStr( dwDestUnreachs
) );
      ICMPIn.Add( ' Час перевищений          : '  + IntToStr( dwTimeEcxcds ) );
      ICMPIn.Add( ' Проблеми з параметрами      : '  + IntToStr( dwParmProbs )
);
      ICMPIn.Add( ' Джерело відключено        : '  + IntToStr( dwSrcQuenchs ) );
      ICMPIn.Add( ' Переназначено            : '  + IntToStr( dwRedirects ) );
      ICMPIn.Add( ' Exo запит        : '  + IntToStr( dwEchos ) );
      ICMPIn.Add( ' Exo відповідь          : '  + IntToStr( dwEchoReps ) );
      ICMPIn.Add( ' Запит мітки часу   : '  + IntToStr( dwTimeStamps ) );
      ICMPIn.Add( ' Відповідь мітки часу    : '  + IntToStr( dwTimeStampReps )
);
      ICMPIn.Add( ' Запит маски адрес : '  + IntToStr( dwAddrMasks ) );
      ICMPIn.Add( ' Відповідь маски адрес   : '  + IntToStr( dwAddrReps ) );
    end;
     //
    with ICMPStats.OutStats do
    begin
      ICMPOut.Add( ' Повідомлення вдправлено        : '  + IntToStr( dwMsgs ) );
      ICMPOut.Add( 'Помилка                 : '  + IntToStr( dwErrors ) );
      ICMPOut.Add( 'Розташування недосягенно    : '  + IntToStr( dwDestUnreachs
) );
      ICMPOut.Add( 'Час перевищений        : '  + IntToStr( dwTimeEcxcds ) );
      ICMPOut.Add( 'Проблеми з параметрами      : '  + IntToStr( dwParmProbs )
);
      ICMPOut.Add( 'Джерело відключено        : '  + IntToStr( dwSrcQuenchs ) );
      ICMPOut.Add( 'Переназначено            : '  + IntToStr( dwRedirects ) );
      ICMPOut.Add( 'Exo запит        : '  + IntToStr( dwEchos ) );
      ICMPOut.Add( 'Exo відповідь          : '  + IntToStr( dwEchoReps ) );
      ICMPOut.Add( 'Запит мітки часу   : '  + IntToStr( dwTimeStamps ) );
      ICMPOut.Add( 'Відповідь мітки часу    : '  + IntToStr( dwTimeStampReps )
);
      ICMPOut.Add( 'Запит маски адрес: '  + IntToStr( dwAddrMasks ) );
      ICMPOut.Add( 'Відповідь маски адрес   : '  + IntToStr( dwAddrReps ) );
    end;
  end
```

```
  else
    IcmpIn.Add( SysErrorMessage( ErrorCode ) );
  Dispose( ICMPStats );
end;

//-----------------------------------
procedure Get_RecentDestIPs( List: TStrings );
begin
  if Assigned( List ) then
    List.Assign( RecentIPs )
end;

initialization

  RecentIPs := TStringList.Create;

finalization

  RecentIPs.Free;

end.
```

Файл About.pas - довідка

```
unit About;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls;

type
  TForm1 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Button1: TButton;
    Image2: TImage;
    Image1: TImage;
    Image3: TImage;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
begin
Form1.Close;
end;

end.
```